

Thesis submitted to the
UNIVERSITY OF MOHAMED BOUDIAF – MSILA



FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in Computer science

By

Bentaleb Khadidja

Bakour Khaoula

Title of the thesis

Peer-To-Peer Delivery Platform

Under the supervision of

Tahar Mehenni

Composition of the jury

S. Hamani

University of Msila

President

Tahar Mehenni

University of Msila

Reporter

Belkacem Brahimi

University of Msila

Examiner

June, 2026

الملخص

شهد مجال خدمات التوصيل في السنوات الأخيرة تطورًا كبيرًا بفضل الاعتماد المتزايد على التقنيات الرقمية وأنظمة التحديد الجغرافي، مما ساهم في تحسين سرعة الخدمات ودقتها وتسهيل عمليات التواصل بين مختلف المستخدمين. وقد أدى هذا التطور إلى ظهور تطبيقات ذكية تهدف إلى تحسين إدارة عمليات التوصيل وتوفير حلول أكثر مرونة وفعالية. تركز هذه المذكرة على تصميم وتطوير تطبيق ذكي خاص بخدمات التوصيل يعتمد على التتبع الجغرافي في الزمن الحقيقي، حيث يوفر منصة تفاعلية تربط بين العميل، السائق، والمستلم. تم تطوير النظام باستخدام مجموعة من التقنيات الحديثة، أبرزها React Native في الواجهة الأمامية، و Node.js مع Express.js في الواجهة الخلفية، بالإضافة إلى PostgreSQL لإدارة قاعدة البيانات، و OpenStreetMap لخدمات الخرائط وتحديد المسارات. كما يوفر التطبيق عدة وظائف مهمة، مثل إرسال طلبات التوصيل، تتبع السائق لحظيًا، حساب أقصر المسارات، إدارة الطلبات، والإشعارات الفورية، مع دعم تعدد اللغات لتحسين تجربة المستخدم. يهدف هذا العمل إلى تقديم نظام ذكي وفعال يساهم في تحسين خدمات التوصيل وتسهيل التفاعل بين مختلف الأطراف داخل النظام.

الكلمات المفتاحية:

خدمات التوصيل، التحديد الجغرافي الديناميكي، التتبع في الزمن الحقيقي

Abstract

The delivery services sector has experienced significant growth in recent years due to the increasing adoption of digital technologies and geolocation systems, which contributed to improving service speed, accuracy, and communication between different users. This evolution has led to the emergence of smart applications aimed at improving delivery management and providing more flexible and efficient solutions.

This thesis focuses on the design and development of a smart delivery application based on real-time geolocation tracking, providing an interactive platform that connects the client, driver, and recipient. The system was developed using modern technologies, mainly React Native for the front-end, Node.js with Express.js for the back-end, PostgreSQL for database management, and OpenStreetMap and OpeRouteService for mapping and route calculation services. The application also provides several important functionalities, such as sending delivery requests, real-time driver tracking, shortest route calculation, request management, and instant notifications, while supporting multilingualism to improve user experience.

This work aims to provide an intelligent and efficient system that contributes to improving delivery services and facilitating interaction between the different actors within the system.

Keywords:

Delivery Services, Dynamic Geolocation, Real-Time Tracking

Résumé

Le domaine des services de livraison a connu un grand développement ces dernières années grâce à l'utilisation croissante des technologies numériques et des systèmes de géolocalisation, ce qui a contribué à améliorer la rapidité des services, leur précision ainsi que la communication entre les différents utilisateurs. Cette évolution a conduit à l'apparition d'applications intelligentes visant à améliorer la gestion des opérations de livraison et à fournir des solutions plus flexibles et efficaces.

Ce mémoire porte sur la conception et le développement d'une application intelligente de livraison basée sur la géolocalisation en temps réel, offrant une plateforme interactive reliant le client, le chauffeur et le destinataire. Le système a été développé à l'aide de technologies modernes, notamment React Native pour le front-end, Node.js avec Express.js pour le back-end, PostgreSQL pour la gestion de la base de données, ainsi que OpenStreetMap et OpenRouteService pour les services de cartographie et de calcul d'itinéraires. L'application offre également plusieurs fonctionnalités importantes, telles que l'envoi des demandes de livraison, le suivi du chauffeur en temps réel, le calcul des plus courts trajets, la gestion des demandes et les notifications instantanées, tout en prenant en charge le multilinguisme afin d'améliorer l'expérience utilisateur.

Ce travail vise à proposer un système intelligent et efficace contribuant à l'amélioration des services de livraison et à la facilitation de l'interaction entre les différents acteurs du système.

Mots-clés :

Services de livraison, Géolocalisation dynamique, Suivi en temps réel.

Dedications

(وَأَخِرُ دَعْوَاهُمْ أَنْ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ)

الحمد لله الذي بنعمته تتم الصالحات، وبفضله تدرك الغايات، وبعونه تذلل الصعوبات،
وبتوفيته تتحقق الأمنيات، لك الحمد حتى ترضى، ولك الحمد اذا رضيت، ولك الحمد بعد
الرضا.....

هذه اللحظة لا تورث ولا تشتري، لقد دفعت ثمنها عمري وشبابي، سهري ودموعي، هي لهفة
الوصول، هذه هي النهايات السعيدة لبدايات أعظم.... أهدي فرحة عمري:

إلى أبي الغالي و أمي الغالية

يا أغلى ما أملك، ويا من لولاهما لما كنت شيئاً مذكوراً.. خذوا هذه الشهادة، فهي ملككم،
وتيجان الفخر هي حق لرؤوسكم... ابنتكم اليوم لا تحتفل بالتخرج، بل تحتفل بأنها ابنة
لعظيمين مثلكما.. ومهما بلغت من عبارات لن أستطيع أن أوفي حقكما العظيم علي.. بارك الله
في عمركما، أدامكما الله ملوكا في عرش قلبي، وحفظكما ملاذا اليه أنتمي...

إلى إخوتي الأعزاء

عبد الصمد وعبد الحق، بلقاسم وزكرياء، قرة عيني ويا من شدد الله بهم عضدي شكراً
لوجودكم الدائم بجانبني، ولكل لحظة محبة منحتهموني إياها.

إلى أختي الغاليتين

فلذتا كبدي ورفيقتا عمري ونجمتاي في الظلام، أنتما السند حين أميل والابتسامة حين أتعب،
أدامكما الله نعمة لي... اكرام وهناء وزوجها جلال الذي أصبح فردا من العائلة.

إلى عائلتي الكبيرة، التي كانت بجانبني طوال عمري، حفظكم الله لي....

وإلى صديقتي الغاليات، رفيقات المشوار والذكريات الجميلة، شكراً لكل لحظة دعم وتشجيع
ومساندة، فقد كنتن جزءاً جميلاً من هذه الرحلة.

بعض المشاعر لا تقال بصوت عال، يكفي ان نجد من يشعر بها معنا بصمت.

إلى كل من دعمني ولو بكلمة طيبة، أهديكم هذا النجاح بكل حب وامتنان.

بكور خولة

(وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ)

صدق الله العظيم

الحمد لله الذي وفقني وأعانني على إتمام هذا العمل، فله الحمد أولاً وآخرًا على نعمه التي لا تُعد ولا تُحصى، وعلى القوة والصبر اللذين منحني إياهما طوال هذه الرحلة
أهدي ثمرة جهدي إلى نفسي، التي صبرت واجتهدت ولم تستسلم رغم كل الصعوبات، فكان
الإصرار طريقها للوصول

إلى أمي الحبيبة، مصدر الحنان والدعاء، التي كانت دائمًا النور الذي يضيء طريقي، شكرًا
لكِ على حبكِ وصبركِ ودعمكِ اللامحدود

إلى أبي العزيز، سندي وفخري، الذي علّمني معنى الاجتهاد والثقة بالنفس، شكرًا لكِ على كل
ما قدمته لي من تشجيع وتضحيات

إلى أخوي العزيزين خالد ورمزي، رفقاء العمر والسند الجميل، شكرًا لوجودكما الدائم إلى
جانبي ولكل لحظات الدعم والمحبة

إلى كل من ساندني ولو بكلمة طيبة، أهدي هذا العمل بكل حب وامتنان

بن طالب خديجة

Acknowledgments

الحمد لله الذي بنعمته تتم الصالحات، والذي وفقنا لإتمام هذا العمل.

يسعدنا في مستهل هذا المقام أن نتقدّم بخالص عبارات الشكر وعظيم الامتنان إلى أستاذنا المشرف، الذي لم يدّخر جهدًا في توجيهنا ومرافقتنا طيلة مراحل إنجاز هذا البحث، بما قدّمه من إرشادات علمية قيّمة، وملاحظات دقيقة، ومتابعة مستمرة، كان لها بالغ الأثر في تحسين جودة هذا العمل والارتقاء به إلى المستوى المنشود.

ولا يسعنا إلا أن نتقدّم بخالص الشكر والامتنان إلى السادة أعضاء لجنة المناقشة الموقّرين، على تفضّلهم بقراءة هذا العمل وتقييمه، وعلى ما سيقدّمونه من ملاحظات قيّمة وتوجيهات بناءة من شأنها إثراء هذا البحث وتعزيز قيمته العلمية.

كما نعبر عن عميق امتناننا لكل من مدّ لنا يد العون وساهم، من قريب أو بعيد، في إنجاز هذا العمل، سواء بالدعم المعنوي أو بالمشورة العلمية أو بالنقاشات المثمرة التي كان لها الأثر في تذليل الصعوبات وتوسيع آفاق هذا البحث.

وفي الختام، نخصّ بالشكر كل من ساندنا ووقف إلى جانبنا خلال هذه المسيرة، راجين من الله أن يجزيهم خير الجزاء.

Contents

General Introduction	1
Chapter1: Automated Delivery and Geolocation Technologies	3
1. Introduction	3
2. Basic Concepts in Delivery	3
2.1 Definition of a Delivery Office (Courier/Delivery Service).....	3
2.2 Operating Method	4
2.2.1 Order Reception Stage	4
2.2.2 Order Organization Stage	4
2.2.3 Manual Route Planning Stage.....	4
2.2.4 Execution and Delivery Stage.....	5
2.2.5 Monitoring and Follow-up Stage.....	5
3. Challenges Faced by Delivery Offices.....	5
3.1 Inaccurate Address Information	5
3.2 Traffic Congestion	6
3.3 Difficulty in Parcel Tracking.....	6
3.4 Inefficient Order Management and Organization.....	6
3.5 Limited Pricing Flexibility.....	6
4. Concept of Geolocation	7
5. Mechanism of Geolocation.....	7
5.1 Global Positioning System (GPS).....	7
5.1.1 Definition	7
5.1.2 How It Works.....	7
5.2 Wi-Fi Positioning System.....	8
5.3 Bluetooth Beacons	8
5.4 Cell Tower Triangulation	8
6. Types of Geolocation	9
6.1 Static Geolocation	9
6.2 Dynamic Geolocation	9
7. Difference Between Static and Dynamic Geolocation	9
7.1 Nature of Movement.....	9
7.2 Data Flow	9
7.3 Applications	10
8. Proposed Solution	10
9. Definition of the Proposed Project.....	10

10. Project Objectives.....	10
11. Conclusion.....	12
Chapter 2: System Analysis and Design	13
1. Introduction	13
2. Proposed System Description	13
3. Requirements Analysis.....	14
3.1 Functional Requirements.....	14
3.2 No-Functional Requirements	15
4. Unified Modeling Language (UML)	15
4.1 Use Case Diagram.....	16
4.2 Sequence Diagram	21
4.3 Class Diagram.....	27
5. Conclusion.....	31
Chapter 3: Development and Implementation.....	32
1. Introduction	32
2. Used Technologies	32
2.1 Development Environment.....	32
2.1.1 Hardware Environment.....	32
2.1.2 Software Environment	33
2.2 Development Stack.....	33
2.2.1 Front-end Technologies.....	33
2.2.2 Back-end Technologies.....	34
2.2.3 Development Tools	34
3. Presentation of the System.....	35
3.1 App Logo	35
3.3 Create Account page	36
3.3.1 Profile of client and driver page.....	39
3.4 Create a new request page.....	42
3.5 My Requests page	42
3.6 Home and Map page	43
3.7 Order Delivery Lifecycle.....	45
3.7.1 Pickup Stage.....	45
3.7.2 Delivery Stage	46
3.8 Tracking and Communication	48
3.9 Cancel Request.....	49
3.10 Emergency Situation	50

4.Application source code	50
5. Conclusion.....	53
General Conclusion	54
References.....	55

List of figures

Fig. 1. 1: How GPS works?.....	8
Fig. 3. 1: App Logo.....	35
Fig. 3. 2: Login Page.....	36
Fig. 3. 3: Verification page	37
Fig. 3. 4: Create account page-Client-	38
Fig. 3. 5: Create account page-Driver-.....	39
Fig. 3. 6: Profile of client	40
Fig. 3. 7: Profile of Driver.....	40
Fig. 3. 8: Verification Code	41
Fig. 3. 9: App Description.....	41
Fig. 3. 10: Create a new request.....	42
Fig. 3. 11: My Request-Client-	43
Fig. 3. 12: Home & Map -Driver-.....	44
Fig. 3. 13: Notification Center	44
Fig. 3. 14: Pickup Stage-Driver-	45
Fig. 3. 15: Pickup Stage-Client-.....	46
Fig. 3. 16: Delivery Stage-Driver-	47
Fig. 3. 17: Delivery Stage-Receiver-	48
Fig. 3. 18: Tracking & call (Driver).....	48
Fig. 3. 19: Notification (pickup & delivery) Stages.....	49
Fig. 3. 20: Cancel request(Client)	49
Fig. 3. 21: Emergency Situation.....	50
Fig. 3. 22: Fetch request.....	51
Fig. 3. 23: Update location the driver (Tracking)	51
Fig. 3. 24: Database connection.....	52
Fig. 3. 25: Create a request and search for a driver.....	52

General Introduction

With the rapid advancement of digital technology, smart applications have become an essential part of daily life, contributing to the transformation of many sectors, especially those related to services and interactions between individuals. The delivery sector is considered one of the most prominent fields that has undergone significant transformation due to the increasing demand for speed, accuracy, and easy access to services.

This project is part of this digital transformation, as many modern systems seek to exploit geolocation technologies in order to improve delivery services. Instead of relying on traditional methods characterized by slowness and limited flexibility, it has become possible to develop intelligent solutions based on direct connection between users and drivers, which contributes to accelerating operations and improving their efficiency. Despite the importance of delivery services, traditional systems still suffer from several shortcomings, including inaccurate location identification, difficulty in tracking orders, and slow response times. Hence, the main problem addressed in this project emerges:

How can an intelligent system be designed to effectively connect clients and drivers while ensuring fast delivery, accurate tracking, and improved service quality?

This project aims to design and develop a smart mobile application dedicated to facilitating delivery operations between individuals, based on real-time geolocation technologies. It seeks to enable users to create delivery requests easily and automatically connect them with the nearest available driver, while also providing real-time order tracking, improving service quality through a rating system, and offering a smooth and efficient user experience.

This project is organized into three interconnected chapters:

- **Chapter One** is devoted to the theoretical study, presenting the fundamental concepts related to delivery services, logistics, and geolocation technologies, along with an overview of the proposed project.
- **Chapter Two** focuses on system analysis and design, where the functional and non-functional requirements are identified, the different actors within the system are studied, and the system's behavior and structure are modeled using UML diagrams such as use case diagrams, sequence diagrams, and class diagrams.

- **Chapter Three** is dedicated to the implementation and development phase, where the system is practically developed, the main tools and technologies used are presented, and the developed interfaces and application workflow are explained.

Finally, this project represents an attempt to provide a practical and simple solution for improving delivery services by exploiting the possibilities offered by modern technologies, with the aim of building an efficient and scalable system.

Chapter1: Automated Delivery and Geolocation Technologies

1. Introduction

The delivery sector is considered one of the vital sectors in daily life, as it plays a major role in facilitating commercial and personal transactions and in transporting goods and parcels between senders and recipients. With the increasing volume and diversity of orders, numerous delivery offices have emerged, operating under specific administrative systems and regulations that often rely heavily on human intervention. This reliance limits delivery speed, flexibility, and the ability to adapt to rapid changes in large urban areas.

In contrast, technological advancement and the emergence of geolocation technology have significantly improved the performance of such systems. Through enabling real-time parcel tracking and optimizing route management, these technologies have enhanced operational efficiency. Scientific studies have demonstrated that integrating digital technologies with delivery systems increases efficiency and reduces risks related to theft and delays.

This chapter aims to provide a comprehensive study of delivery offices, with a focus on analyzing the role of geolocation in enhancing transport and delivery services. It also prepares the transition toward a new delivery model based on direct citizen participation, ensuring greater flexibility, higher efficiency, and improved security in delivery operations.

2. Basic Concepts in Delivery

2.1 Definition of a Delivery Office (Courier/Delivery Service)

A delivery office is defined as a logistics system specialized in transporting parcels and goods from the point of dispatch to the point of reception (sender–recipient) in an organized and efficient manner. Its objective is to meet customer needs as quickly as possible while maintaining high levels of accuracy and reliability. Delivery offices represent a fundamental component of modern supply chains, particularly with the growth of e-commerce and the increasing reliance on fast delivery services [1].

2.2 Operating Method

The operational mechanism of traditional delivery offices relies largely on manual administrative and organizational procedures throughout most stages of the process. This model is based on direct coordination between employees and drivers using communication tools such as telephone calls and paper records. The process typically involves interconnected stages as follows:

2.2.1 Order Reception Stage

The process begins with receiving delivery requests from customers through:

- Direct visits to the office;
- Phone calls ;
- Short messages or basic communication tools.

The responsible employee manually records the order details in a paper register or a simple electronic table. The recorded data usually includes:

- Sender and recipient names ;
- Delivery address (textual description) ;
- Phone number ;
- Parcel type and weight ;
- Cash-on-delivery amount (if applicable).

In this traditional model, textual address descriptions are used without precise geographic coordinates, which may lead to errors or delivery delays.

2.2.2 Order Organization Stage

After collecting the orders, the office manager proceeds to:

- Group parcels according to general geographic areas (districts, municipalities, cities);
- Manually prioritize them based on reception time;
- Prepare delivery lists for each driver.

Parcel distribution among drivers is typically based on personal experience or prior knowledge of the area rather than route optimization algorithms.

2.2.3 Manual Route Planning Stage

In the traditional system, drivers rely on:

- Personal knowledge of road networks ;
- Paper maps or basic navigation applications;

- Phone guidance from customers when necessary.

There is no centralized system to calculate the shortest or fastest route; instead, route selection depends on personal judgment.

2.2.4 Execution and Delivery Stage

The driver delivers the parcels according to the assigned list. Upon arrival:

- The driver verifies the recipient's identity;
- Collects payment in cash (if applicable);
- Obtains the recipient's signature in a paper register as proof of delivery

If the recipient is unavailable, the parcel is returned to the office or rescheduled for another delivery attempt.

2.2.5 Monitoring and Follow-up Stage

In the traditional mechanism:

- There is no real-time tracking of drivers;
- Drivers are contacted by phone to determine their location;
- Order statuses are manually updated at the end of the day.

Performance reports are prepared based on paper records, which may result in inaccuracies or difficulties in generating detailed statistics.

3. Challenges Faced by Delivery Offices

Despite the significant role that delivery offices play in facilitating the transportation of parcels and goods between individuals and businesses, this sector faces several challenges that directly impact the efficiency and quality of the services provided. These challenges become more evident with the increasing demand for delivery services and the rapid expansion of urban areas. The following are the main difficulties encountered by traditional delivery systems.

3.1 Inaccurate Address Information

One of the most common challenges in traditional delivery services is the lack of precise address information. In many cases, the location of the recipient is determined based on textual descriptions such as the name of the neighborhood, street, or a nearby landmark, without relying on accurate geographic coordinates.

This method often makes it difficult for delivery workers to identify the exact destination, forcing them to spend additional time searching for the address or contacting the recipient

multiple times. Consequently, this issue can lead to delivery delays and increased operational costs.

3.2 Traffic Congestion

Traffic congestion is another major factor affecting the efficiency of delivery services, particularly in densely populated urban areas. Heavy traffic conditions increase travel time between delivery points, which significantly reduces the number of deliveries that can be completed within a given timeframe.

Furthermore, traffic congestion leads to higher fuel consumption and negatively affects the productivity of delivery personnel, thereby increasing the overall operational costs for delivery companies.

3.3 Difficulty in Parcel Tracking

Many traditional delivery offices rely on limited monitoring methods, such as telephone communication, to obtain information about the status of an order or the location of the delivery driver. The absence of digital tracking systems makes it difficult to determine the exact location of a parcel during the delivery process. Additionally, the lack of real-time tracking reduces transparency between the delivery service provider and the customer. As a result, customers are often unable to monitor the progress of their deliveries unless they directly contact the delivery office or the driver [1].

3.4 Inefficient Order Management and Organization

In many delivery offices, orders are managed manually or through simple tools such as paper records or basic electronic spreadsheets. This traditional approach can make it difficult to efficiently handle a large number of orders simultaneously, especially when demand increases.

Moreover, poor organization and manual data entry may result in errors when recording information or assigning deliveries to drivers. These issues may lead to delivery delays, operational inefficiencies, or even the loss of certain orders[2].

3.5 Limited Pricing Flexibility

Another challenge faced by many delivery offices is the lack of flexibility in pricing policies. Most traditional delivery services rely on fixed pricing structures based on predefined geographic zones or estimated distances.

Although this pricing model simplifies the process of calculating delivery costs, it does not always reflect the actual conditions of the delivery process. In some cases, the delivery fee may be relatively high compared to the actual distance or the value of the parcel, while in other cases it may not adequately compensate the delivery worker for the effort involved.

Furthermore, the absence of negotiation mechanisms or dynamic pricing strategies limits the system's ability to adapt to changes in demand, traffic conditions, or operational costs.

4. Concept of Geolocation

Geolocation refers to the process of determining the precise geographic position of a person, vehicle, or device on the Earth's surface in real time using geographic coordinates (longitude, latitude, and sometimes altitude). Longitude and latitude define the exact position of a point on the Earth's surface, while altitude adds a vertical dimension. These coordinates are linked to information systems that allow data analysis and visualization on digital maps [3].

5. Mechanism of Geolocation

Geolocation relies on an integrated set of systems and technologies, mainly:

5.1 Global Positioning System (GPS)

5.1.1 Definition

GPS is a satellite-based navigation system that allows real-time positioning of objects. In business contexts, it is primarily used for tracking company vehicles, enhancing security (especially against theft or for emergency assistance), and analyzing driving behavior to improve operational performance [3].

5.1.2 How It Works

GPS operates as follows:

- The GPS device receives signals from at least three satellites to calculate location using the triangulation principle;
- The embedded processing unit converts these signals into GPS coordinates and may associate additional data (speed, stopping time, etc.);
- A communication unit then transmits the data to a remote server via GSM, GPRS, 3G/4G/5G networks, often using an integrated SIM card. In areas with weak network coverage, some devices use satellite links to update information[3].

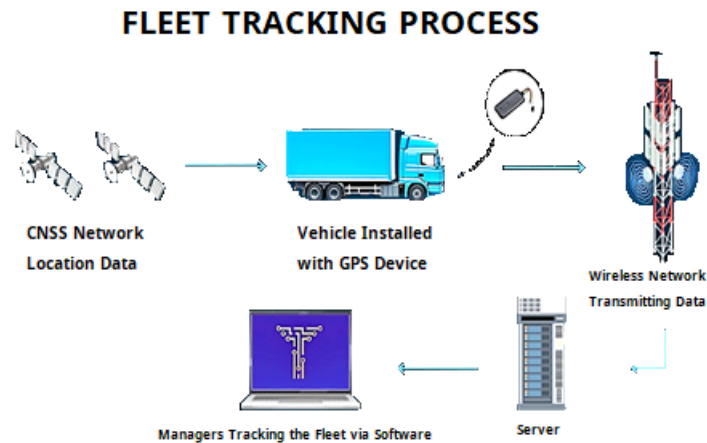


Fig.1. 1: How GPS works? [3].

5.2 Wi-Fi Positioning System

The Wi-Fi Positioning System (WPS) determines the location of a device by analyzing signals from nearby wireless access points. A smartphone scans the available Wi-Fi networks in its surroundings, and these networks are compared with a database containing the geographic coordinates of known access points.

Through this process, the system can estimate the device's location with reasonable accuracy, particularly in urban areas where Wi-Fi networks are widely available. This technique is commonly used in mobile applications, especially in environments where GPS signals are weak, such as inside buildings or tunnels [4].

5.3 Bluetooth Beacons

Bluetooth Beacons are small devices that transmit Bluetooth Low Energy (BLE) signals continuously. When a smartphone enters the range of a Beacon, a mobile application can detect the signal and estimate the user's location based on the signal strength.

This technology is widely used for indoor positioning in environments such as shopping malls, airports, and large buildings, where traditional GPS signals may not function effectively [6].

5.4 Cell Tower Triangulation

Cell Tower Triangulation is a technique used to estimate the location of a mobile device by analyzing signals received from nearby cellular network towers. When a smartphone connects to the mobile network, it communicates with one or more cell towers. By measuring the signal strength and the distance between the device and multiple towers, the system can estimate the approximate geographic position of the device.

Although this method is generally less accurate than GPS, it remains useful in areas where GPS signals are weak or unavailable, such as urban environments or indoor locations [5].

6. Types of Geolocation

Geolocation is scientifically classified according to the nature of movement, data flow pattern, and coordinate update mechanism. The most commonly adopted classification distinguishes between static geolocation and dynamic geolocation.

6.1 Static Geolocation

This type typically relies on GPS systems. Technically, it involves receiving signals from at least three satellites and calculating coordinates using triangulation algorithms. Since the object is stationary, the focus is on improving absolute accuracy rather than tracking movement [3].

6.2 Dynamic Geolocation

Dynamic geolocation refers to continuously or periodically determining and tracking the position of a moving object, with coordinate updates in real time or at short intervals.

In this context, the process extends beyond calculating a single geographic point to recording routes, calculating speed and direction, estimating expected arrival time, and analyzing movement behavior [3].

7. Difference Between Static and Dynamic Geolocation

7.1 Nature of Movement

Static geolocation relates to non-moving objects with stable coordinates over time, whereas dynamic geolocation deals with moving objects whose coordinates constantly change and therefore require periodic updates.

7.2 Data Flow

In the static model, data flow is limited and occasional. In the dynamic model, data transmission is continuous and depends on network connectivity for real-time analysis.

7.3 Applications

Static applications are mainly referential or documentary (mapping, surveying), while dynamic applications are operational and interactive (smart transportation, delivery services).

8. Proposed Solution

After reviewing traditional delivery systems and geolocation technologies, it becomes clear that current technological capabilities allow the development of more flexible solutions based on direct interaction between individuals through mobile applications that rely on real-time geographic positioning. From this perspective emerges the proposed project idea.

9. Definition of the Proposed Project

The idea of this project is to design and develop a smart mobile application that functions as an interactive digital platform aimed at facilitating parcel delivery between individuals in a fast and efficient manner. The application connects two main categories of users within a unified digital environment:

- Individuals who wish to send a parcel or an item to another person located in a different geographical area;
- Independent drivers who own various means of transportation such as cars, motorcycles, or bicycles and are willing to provide delivery services in exchange for financial compensation.

The application relies on geolocation technologies to accurately determine the locations of drivers, which enables the system to assign delivery requests to the nearest available driver. In addition, the application provides a real-time tracking feature, allowing users to monitor the delivery process and follow the parcel's route from the moment it is picked up by the driver until it is successfully delivered to the recipient.

10. Project Objectives

This project aims to achieve a set of key objectives that contribute to the development of delivery services and enhance their efficiency and flexibility. The main objectives of this project include:

- Facilitating the process of sending parcels between individuals through an easy-to-use digital platform;
- Utilizing geolocation technologies to accurately determine the locations of both users and drivers;
- Reducing the time required for delivery operations by assigning requests to the nearest available driver;
- Providing a real-time tracking system that enables users to monitor the status of their orders step by step;
- Creating flexible job opportunities for individuals who own means of transportation and wish to provide delivery services;
- Improving the efficiency of delivery operations management through the use of modern digital technologies;
- The project aims to provide a flexible and adaptive pricing mechanism that considers multiple factors such as distance, demand, and traffic conditions, ensuring a fair balance between service cost and quality.

11. Conclusion

This chapter demonstrates that delivery systems constitute a fundamental component of the contemporary logistics system, particularly with the expansion of e-commerce and increasing demand for fast transportation services. Despite their economic importance, they face numerous challenges from multiple perspectives.

The chapter also shows that geolocation, as a technology based on satellite navigation systems and real-time geographic data processing, represents a strategic tool capable of reshaping modern delivery systems by improving operational efficiency, reducing costs, and enhancing service transparency

Therefore, transitioning from traditional delivery models to intelligent systems based on geolocation is not merely a technological option but an organizational and economic necessity dictated by modern market demands. This theoretical framework forms the foundation upon which the practical component of the project will be developed in subsequent chapters, aiming to present an innovative and scalable solution that improves delivery services and enhances their effectiveness and sustainability.

Chapter 2: System Analysis and Design

1. Introduction

After presenting the theoretical framework in the first chapter, which addressed the concepts of delivery services, logistics, and geolocation technologies, this chapter represents the practical extension of this work by focusing on the systematic analysis and design of the proposed system. It aims to provide a clear understanding of how the application operates by identifying its requirements and establishing a well-defined vision of its components and operational mechanisms.

This chapter also focuses on identifying the different actors involved in the system, namely the user, the driver, and the receiver, while analyzing the interactions between them in a way that reflects the real workflow of the application. This analysis contributes to clarifying the system's logic and the flow of its processes in a coherent and organized manner.

In this context, UML diagrams are adopted as a fundamental tool for system modeling, through the development of use case diagrams, sequence diagrams, and class diagrams. These diagrams help provide a comprehensive view that facilitates the development process and reduces the likelihood of errors during implementation.

Therefore, this chapter constitutes a key step toward the actual implementation phase by establishing a solid design foundation that ensures the development of an efficient, well-structured, and scalable system.

2. Proposed System Description

The proposed system consists of a smart application designed to facilitate delivery operations between individuals. It aims to connect customers who wish to send parcels or goods with independent drivers willing to provide delivery services in exchange for a negotiable fee, within an interactive digital environment based on geolocation technologies.

The system operates on the principle of intelligent matching between customers and drivers, allowing the customer to enter order details, including the pickup location and delivery destination, after which the system analyzes this data and assigns the request to the nearest available driver based on real-time location, previous ratings, and vehicle type. Once the driver accepts the request, it must also be confirmed by the customer before the delivery process

begins, after which real-time tracking is provided, enabling both the customer and the receiver to monitor the parcel's route until it is successfully delivered.

In addition, the system offers a flexible payment mechanism and allows users to rate drivers after completing the delivery, which contributes to improving service quality and enhancing trust among users. Overall, the system is characterized by its ability to reduce delivery time, optimize resource utilization, and provide a smooth and efficient user experience.

3. Requirements Analysis

Requirements analysis aims to identify the various functions that the system must provide, in addition to the non-functional characteristics that ensure its performance quality and stability. This analysis represents a fundamental step to ensure that the system aligns with user needs and achieves the project objectives.

3.1 Functional Requirements

Functional requirements represent the set of operations and services that the system must provide to its users, and can be summarized as follows:

- Allow users to create a new account within the system;
- Enable users to log in using their personal credentials;
- Allow users to submit a delivery request including parcel details, pickup location, and delivery destination;
- Assign the request to the nearest available driver based on geolocation and previous rating and the type of vehicle;
- Enable the driver to accept or reject delivery requests;
- Allow the client to accept or reject the assigned driver;
- Execute the delivery process from the pickup location to the delivery destination;
- Provide real-time tracking of the delivery request;
- Enable the recipient to confirm the receipt of the parcel;
- Provide a flexible payment system based on negotiation;
- Allow both the client and the recipient to rate the driver after the delivery is completed.

3.2 No-Functional Requirements

No-functional requirements define the quality attributes and performance of the system, including:

1. Performance:

The system must respond quickly to user requests, especially for tracking and request assignment operations.

2. Security:

Ensure the protection of user data through authentication mechanisms.

3. Usability:

Provide a simple and user-friendly interface that allows easy interaction without complexity.

4. Availability:

Ensure continuous system operation without interruption.

5. Scalability:

The system should be capable of handling a large number of users and requests efficiently.

6. Pricing Flexibility:

The driver relies on a flexible pricing mechanism that considers multiple factors such as distance and traffic conditions, ensuring a fair balance between service cost and quality.

4. Unified Modeling Language (UML)

set of integrated diagrams and graphical notations. It was developed to help software engineers and system developers analyze, design, visualize, and document information systems. UML allows the representation of system components, relationships, and behaviors in an organized visual manner that facilitates understanding the system structure before the implementation phase.

UML represents a set of engineering best practices that have proven effective in modeling large and complex systems. It is widely used in object-oriented software development and plays an important role in the software development process. UML diagrams help development teams communicate more effectively, analyze system requirements, explore different design solutions, and validate the system architecture before the implementation stage[7].

4.1 Use Case Diagram

A Use Case is formally defined as a structured and systematic description of system behavior from the user's perspective. It represents an organized interaction between an external actor and the system in order to achieve a specific goal that delivers functional value to the user. A use case describes a logical sequence of events that illustrates how the system responds to various inputs, including both normal and exceptional scenarios, making it a fundamental tool for functional requirements analysis and for precisely defining system boundaries and functionalities[7].

The system consists of three main actors: **Client**, **Driver**, and **Receiver**, where each actor has a set of use cases that reflect their roles and functions within the system.

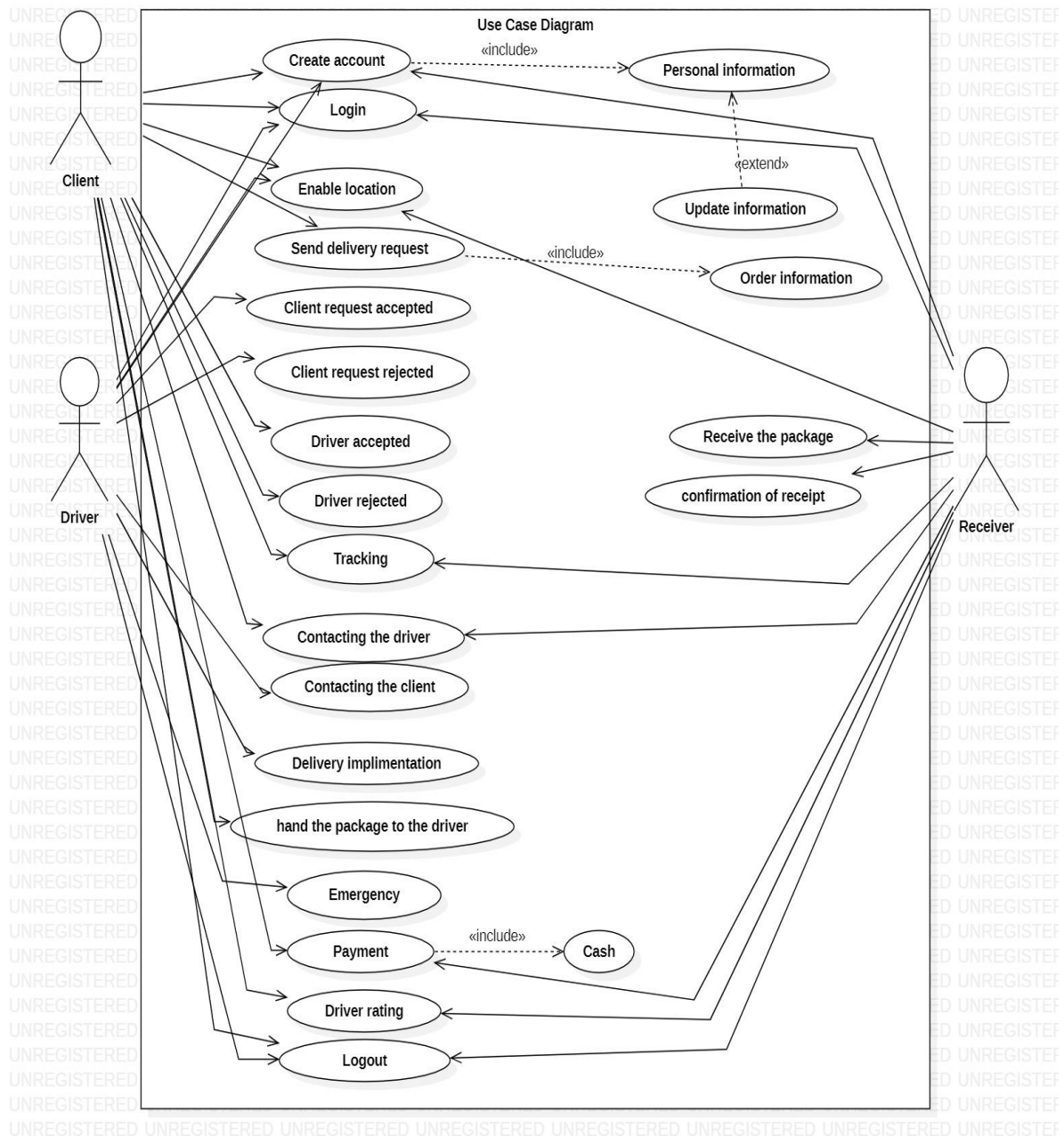


Fig. 2. 1: Use Case Diagram

Account Management

These functionalities represent the common operations shared by all actors in the system and include:

- **Create Account:** This use case allows all actors to create a new account in the system; It has a mandatory relationship with **Personal Information**, meaning that entering personal data is an essential step that cannot be omitted during the registration process.

- **Update Information:** It has an «extend» relationship with Personal Information, which means it is an optional function that allows users to modify their data after creating the account;
- **Login:** Allows users to access the system using their personal credentials;
- **Logout.**

Actor: Client

The client is the main actor in the system, as they initiate the delivery process and follow all its stages.

Use Cases:

- **Enable Location:** Necessary to accurately determine the client's location, allowing the system to properly direct the request;
- **Send Delivery Request:** Represents the core functionality and has a mandatory relationship with **Order Information**, where the user must provide package details and both pickup and delivery locations;
- **Driver Accepted / Rejected:** After the system suggests a driver, the client can accept or reject the proposed driver;
- **Tracking:** Enables the client to monitor the delivery process in real time;
- **Contacting the Driver:** Allows communication during the delivery process;
- **Payment:** Uses the **Cash** method, meaning cash payment is part of the process;
- **Driver Rating:** Enables the client to evaluate the driver after the delivery is completed, in order to improve service quality.

Actor: Driver

The driver represents the executing party of the delivery process, interacting with requests and carrying them out.

Use Cases:

- **Enable Location:** Allows the system to determine the driver's location and assign nearby requests;

- **Client Request Accepted / Rejected:** The driver can accept or reject delivery requests;
- **Delivery Implementation:** Represents the main operation performed by the driver;
- **Hand the Package to the Driver:** Represents the stage where the driver receives the package from the client, marking the actual start of the delivery process;
- **Contacting the Client:** Allows communication with the client to coordinate pickup or delivery;
- **Emergency:** Handles exceptional situations during delivery.

Actor: Receiver

The receiver represents the final actor in the delivery process, where the operation ends.

Use Cases:

- **Enable Location:** Used to determine the receiver's location when using the application;
- **Tracking:** Allows the receiver to follow the delivery status before receiving the package;
- **Contacting the Driver:** Enables communication with the driver to coordinate delivery;
- **Receive the Package:** Represents the final step of the delivery process;
- **Payment:** Based on cash payment if agreed upon;
- **Confirmation of Receipt:** Used to confirm the successful completion of the process;
- **Driver Rating:** Contributes to improving service quality.

This diagram highlights a logical and integrated flow of the delivery process, starting from the client, passing through the driver, and ending with the receiver. It also provides supporting functionalities such as tracking, communication, and payment. This analysis helps in providing a clear understanding of the system's behavior.

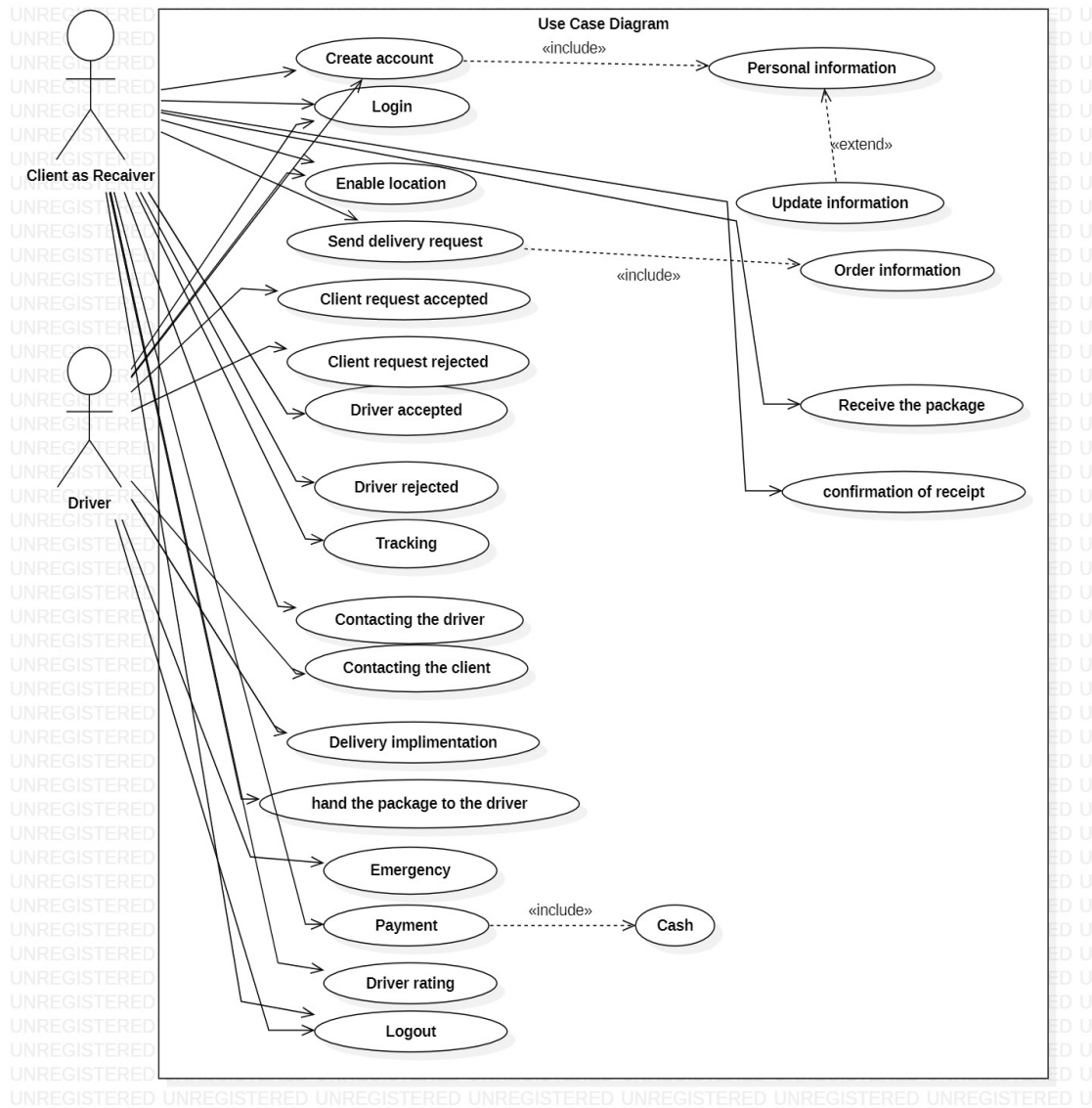


Fig. 2. 2: Special Use Case Diagram

Special Case : Client as Receiver

This scenario represents a special case in the system, where the user assumes both the sender and receiver roles at the same time, which simplifies the process flow without affecting the system structure or its core functionalities.

The process begins with creating a delivery request after enabling location services and entering delivery details. The system then suggests a driver, who can be accepted or rejected by the client. Once an agreement is reached, the delivery process is carried out with real-time tracking and direct communication. Since the client is also the receiver, they receive the package, confirm the operation, complete the payment, and rate the driver.

The importance of this scenario lies in highlighting the system's flexibility and its ability to merge roles, as it maintains the same workflow logic while reducing complexity, thereby enhancing user experience and supporting simpler and more realistic usage scenarios.

4.2 Sequence Diagram

A Sequence Diagram is one of the fundamental interaction diagrams in the Unified Modeling Language (UML). It is used to model the dynamic behavior of a system by representing the time-ordered interactions between objects or system components. This diagram presents objects as vertically arranged lifelines, while the communications between them are depicted as exchanged messages ordered chronologically from top to bottom, reflecting the logical sequence of operations [7].

This sequence diagram represents the complete chronological flow of the delivery process within the system, illustrating the interactions between the different actors: the Client, the System, the Driver, and the Receiver.

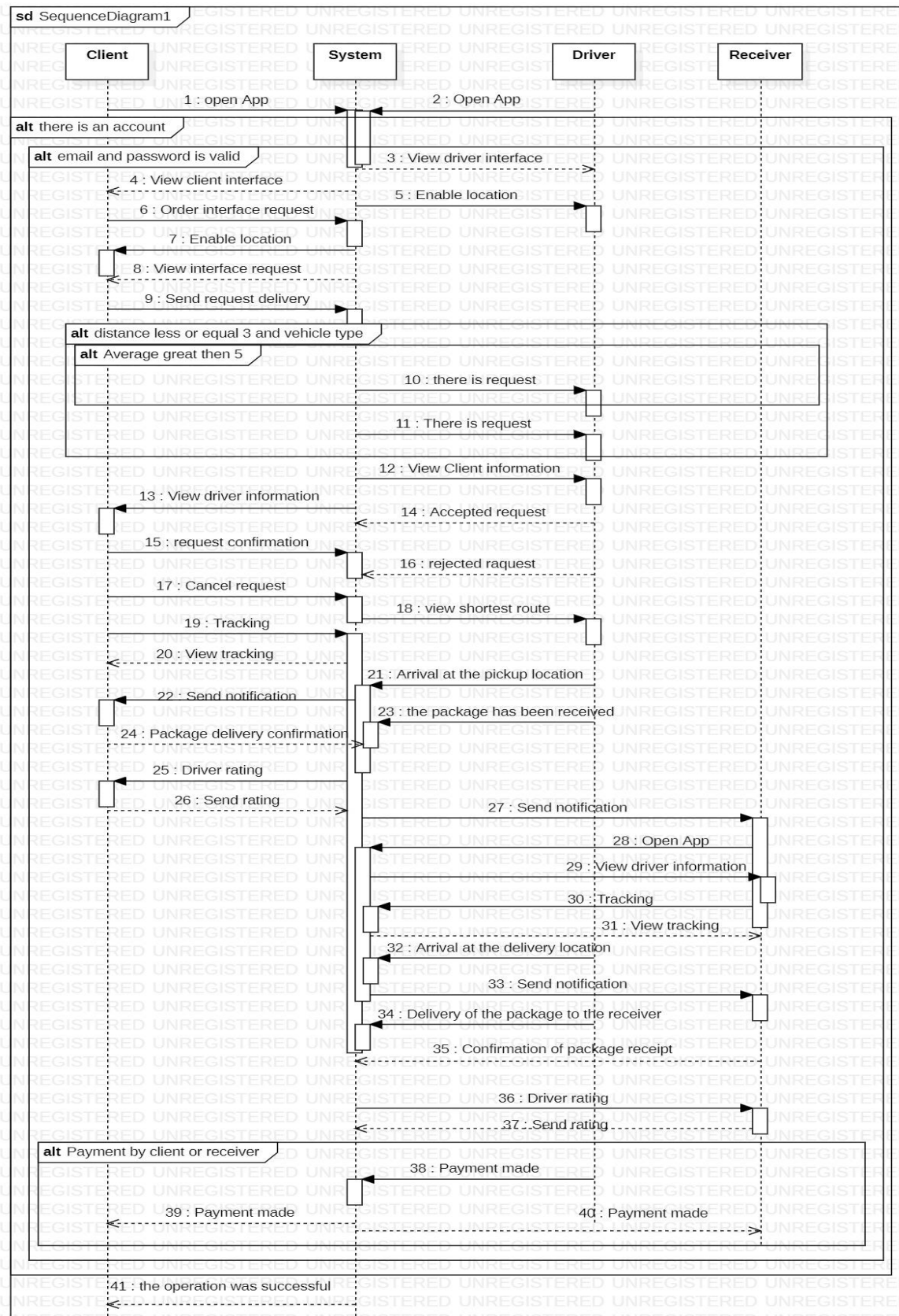


Fig. 2. 3: Sequence Diagram

1. Login and System Initialization Phase

The process begins when both the client and the driver open the application and verify that they already have accounts in the system. Authentication is then performed by checking the email address and password.

Once the login process is successful:

- The client interface dedicated to sending delivery requests is displayed, with location services enabled so the system can determine the client's position and suggest the appropriate driver;
- The driver interface dedicated to receiving delivery requests is also displayed.

The driver then activates geolocation (Enable Location) so that the system can identify the current position and use it during the request assignment process.

2. Delivery Request Creation Phase

The client accesses the request interface and enters the delivery information, including:

- Pickup location ;
- Delivery location ;
- Package details.

After that, the delivery request is sent to the system (Send Request Delivery).

3. Suitable Driver Selection Phase

The system relies on several criteria to select the most suitable driver, mainly:

- The distance between the driver and the pickup location;
- Vehicle type ;
- Driver rating.

This is represented in the diagram using alternative conditions (alt), where the driver is proposed if:

- The distance is less than or equal to 3 km;
- The driver has a good rating.

The system then forwards the request to the selected driver.

4. Request Acceptance or Rejection Phase

When the driver receives the request:

- The driver can view the client's information;
- Then decide whether to accept or reject the request;

In case of rejection:

- The request is canceled or reassigned to another driver;

In case of acceptance:

- The system sends a confirmation notification to the client;
- The actual delivery execution phase begins.

5. Tracking and Delivery Execution Phase

After the request is accepted:

- The system calculates the shortest route for the driver;
- Real-time tracking is activated.

The client can monitor the driver's movement directly through the application, while notifications are sent when:

- The driver arrives at the pickup location;
- The package is received ;
- The driver starts heading toward the delivery location.

6. Receiver Interaction Phase

The system sends a notification to the receiver informing them about the delivery process.

After opening the application:

- The receiver can view the driver's information;
- Track the delivery status in real time.

When the driver arrives at the delivery location:

- The package is delivered to the receiver;
- The receiver confirms the receipt of the package (Confirmation of Package Receipt).

7. Rating and Payment Phase

After the delivery process is completed:

- The client or the receiver can rate the driver (Driver Rating), which contributes to improving service quality.

The diagram also shows an alternative payment scenario (alt), where payment can be made either by:

- The client ;
- Or the receiver.

Finally :

- The payment process is confirmed;
- The system sends a final notification indicating the successful completion of the entire operation.

Special Sequence Diagram

This diagram represents the sequence of interactions exchanged between the client, the system, and the driver during the execution of the delivery process in the special case where the client is also the receiver.

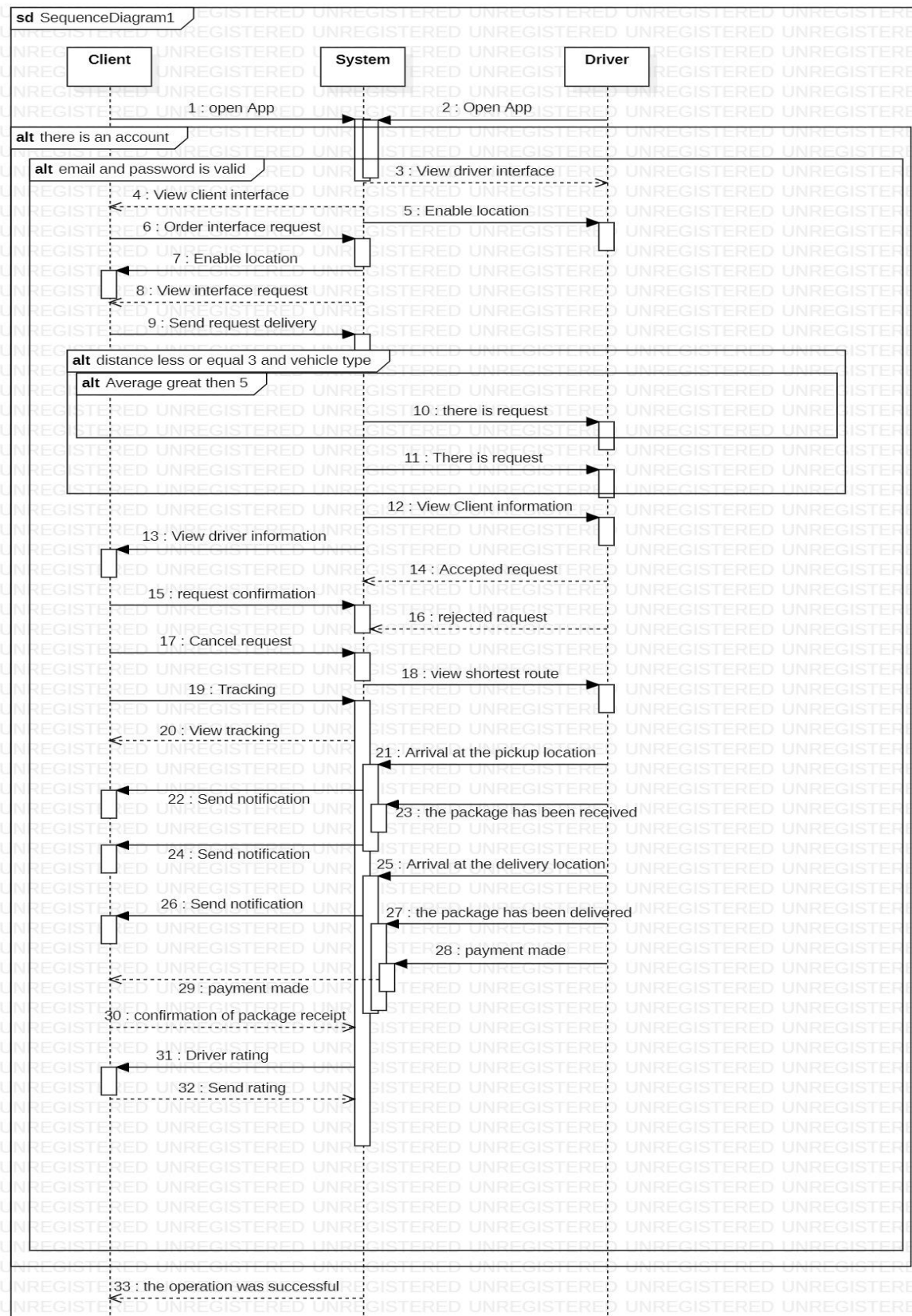


Fig. 2. 4: Special Sequence Diagram

The process begins with logging in and enabling geolocation services, after which the client creates a delivery request and sends the package information, pickup location, and delivery location to the system. The system then analyzes the request and selects the most suitable driver based on distance, vehicle type, and driver rating before sending the request for acceptance or rejection. In the case of acceptance, the delivery process starts with the activation of real-time tracking and instant notifications to monitor the different stages of the operation. Upon the driver's arrival, the client — acting also as the receiver — confirms the receipt of the package, after which the payment process and driver rating are completed. This diagram highlights the integration between the different system components and reflects the application's reliance on geolocation tracking and smart notifications to ensure an efficient and seamless delivery process.

4.3 Class Diagram

A Class Diagram is one of the fundamental structural diagrams in the Unified Modeling Language (UML). It is used to represent the static structure of a system by defining its classes, their attributes, and methods, as well as the relationships that connect them.

The class diagram aims to provide a comprehensive view of the organization of system components from an object-oriented perspective, highlighting how software entities are structured and interact within a coherent logical framework. It is also considered a key tool in the design phase, as it serves as a foundation for transforming the conceptual model into an implementable model, thereby improving design quality and ensuring system clarity, maintainability, and scalability [7].

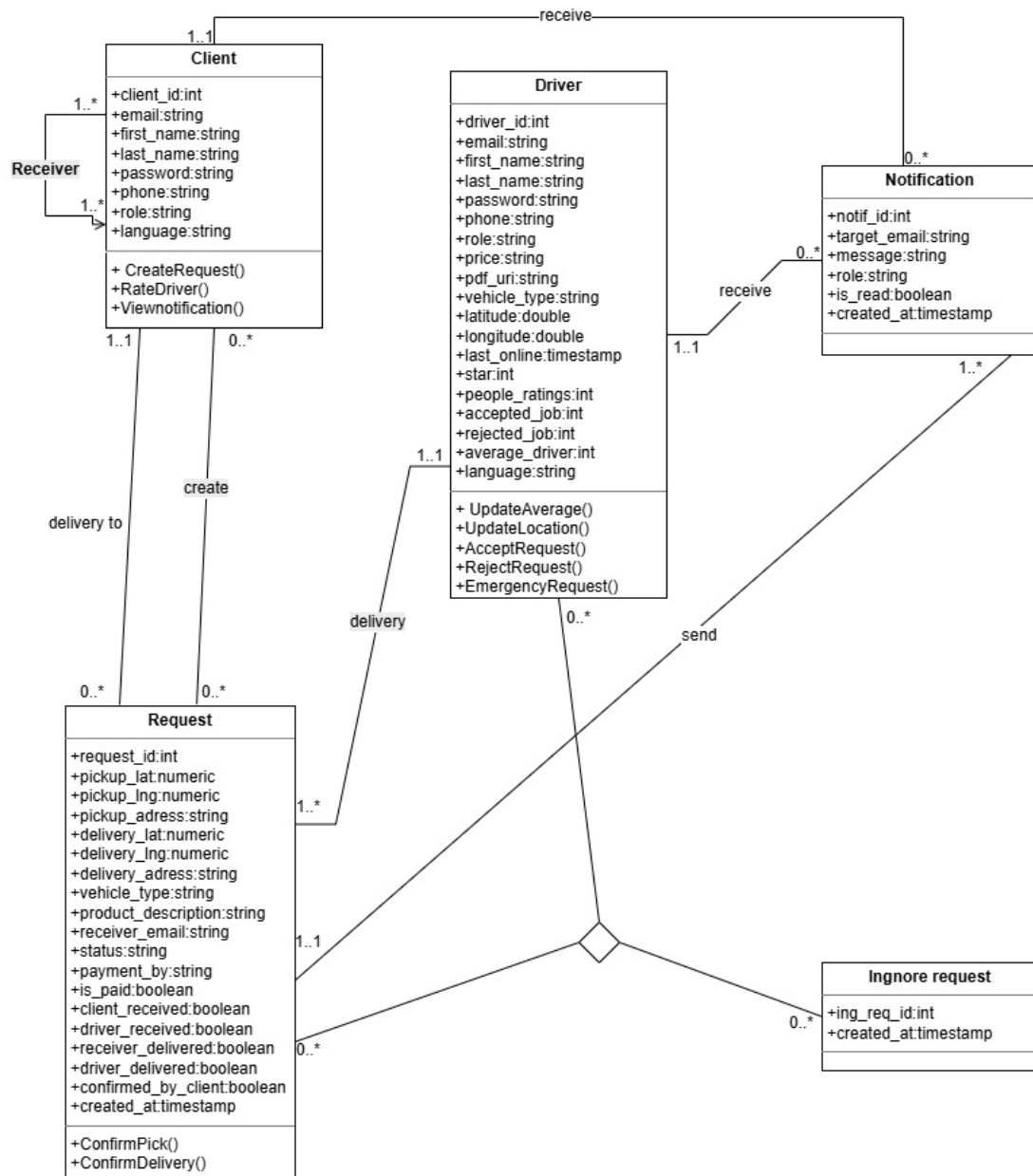


Fig. 2. 5: Class Diagram

Main Entities of the System

1. Client

Represents the user who creates delivery requests within the system. This class contains personal information such as the first name, last name, email address, phone number, and password, role. It also includes several operations that allow the client to interact with the system, including:

- **CreateRequest():** used to create a new delivery request;
- **RateDriver():** allows the client to evaluate the driver after completing the delivery process;
- **Viewnotification():** enables the client to view received notifications.

The Client class is associated with the Request class through a **create** relationship, where one client can create multiple delivery requests, and the **delivery to** relationship applies in the case of delivery to the receiver.

2. Driver

Represents the driver responsible for carrying out the delivery operation. This class contains detailed information such as the geographical location (latitude, longitude), rating score (star), and the number of accepted or rejected requests.

Its main operations include:

- **AcceptRequest() / RejectRequest():** used to manage delivery requests;
- **UpdateLocation():** updates the driver's location in real time;
- **UpdateAverage():** calculates and updates the average rating;
- **EmergencyRequest():** handles emergency situations during delivery.

The Driver class is linked to the Request class through a **delivery** relationship, since one driver can handle multiple requests.

3. Request

This is the core class of the system, representing the delivery request itself. It contains detailed information such as:

- Pickup and delivery locations ;
- Package description ;
- Request status (status) ;
- Payment and delivery confirmation states .

It also provides several operations, including:

- **ConfirmPick()**: confirms package pickup.
- **ConfirmDelivery()** : confirms successful delivery.

This class is associated with both Client and Driver, making it the central link between the different components of the system.

4. Notification

Represents the notification management system within the application. It contains information such as:

- Message content ;
- Read status .

The Notification class is connected to both Client, Receiver and Driver through **send/receive** relationships, where each user can send or receive multiple notifications.

5. IgnoreRequest

Represents the cases in which a driver ignores a specific delivery request, and this contributes to tracking driver behavior.

5. Conclusion

In this chapter, the transition was made from the theoretical aspects discussed in the first chapter to the practical aspects related to the design of the proposed system. The focus was placed on modeling the system using UML diagrams, which provided a clear and structured representation of its different aspects. The Use Case Diagram made it possible to identify the main interactions between actors and the system, while the Sequence Diagram highlighted the dynamic flow of message exchanges over time between its components. In addition, the Class Diagram provided a precise representation of the system's structural organization.

These models collectively contributed to building a comprehensive and coherent view of the system architecture, facilitating its understanding and analysis prior to the implementation phase. Therefore, this chapter constitutes an essential design foundation that paves the way for the development stage and the practical realization of the system.

Chapter 3: Development and Implementation

1. Introduction

After completing the system analysis and design phase in the second chapter, this chapter presents the practical aspect of the project by moving to the actual implementation phase of the proposed application. This chapter represents an essential step in transforming the theoretical models and designs into a real and usable system based on modern technologies that ensure the efficiency and stability of the application.

This chapter aims to present the different tools and technologies adopted during the development process, while explaining the development environment, programming languages, and software applications used in building the application. It also includes a presentation of the main implemented interfaces for the different users of the system, along with an explanation of how they work and how users interact with them. In addition, it highlights some core functionalities such as sending delivery requests, driver tracking, notifications, and delivery management.

2. Used Technologies

This section presents the various software tools and technologies used in the development of the proposed application.

2.1 Development Environment

The application was developed using a set of modern software tools and technologies in order to ensure the development of an efficient, stable, and scalable system. The development environment included both hardware and software aspects, where a computer with suitable specifications was used for developing and running the application, in addition to several software technologies that contributed to implementing the different system functionalities.

2.1.1 Hardware Environment

The application was developed using a computer with the following specifications:

- **Processor:** Intel Core i5 / 8th Gen;
- **RAM:** 16 GB ;
- **Operating System:** Windows 11 Professional ;

- **Storage:** SSD 256 GB .

2.1.2 Software Environment

On the software side, several modern tools and technologies were used to develop the application and implement its different functionalities, including:

- **Visual Studio Code (VS Code):** Used as the development environment for writing and organizing the source code.
- **GitHub:** Used for project management, version control, and sharing project files.
- **React Native:** Used for developing the mobile application and building user interfaces.
- **Expo Go:** Used for testing and running the application during the development phase.
- **Node.js:** Used for developing the back-end part of the application.
- **PostgreSQL:** Used as the database management system for storing system information.
- **OpenStreetMap:** Used for displaying maps and geolocation services.
- **OpenRouteService:** Used for route calculation and determining the shortest delivery path based on the type of vehicle used by the driver.

2.2 Development Stack

2.2.1 Front-end Technologies

- **React Native:** An open-source framework developed by Meta, used for building cross-platform mobile applications using JavaScript while providing performance close to native applications [8].
- **OpenStreetMap:** An open-source mapping project used to create and display geographic maps while providing accurate spatial data that can be used in tracking and navigation applications [9].
- **OpenRouteService:** A routing service based on OpenStreetMap data used to calculate routes, distances, and the shortest paths between different geographic locations [10].
- **HTML:** A markup language used to structure and organize the content of web pages and application interfaces, such as texts, buttons, images, and forms.
- **CSS:** A styling language used to define the visual appearance of application interfaces, including colors, fonts, spacing, and layout organization.

- **JavaScript (JS):** A programming language used to add interactivity and dynamic behavior to the application, such as event handling, data validation, and communication with the server[11].
- **TSX (TypeScript XML):** A syntax extension used with TypeScript and React that allows combining programming code with interface components to build organized and interactive user interfaces efficiently[12].
- **i18next:** An internationalization framework used to support multilingualism within the application by dynamically translating user interfaces and managing content in multiple languages[13].

2.2.2 Back-end Technologies

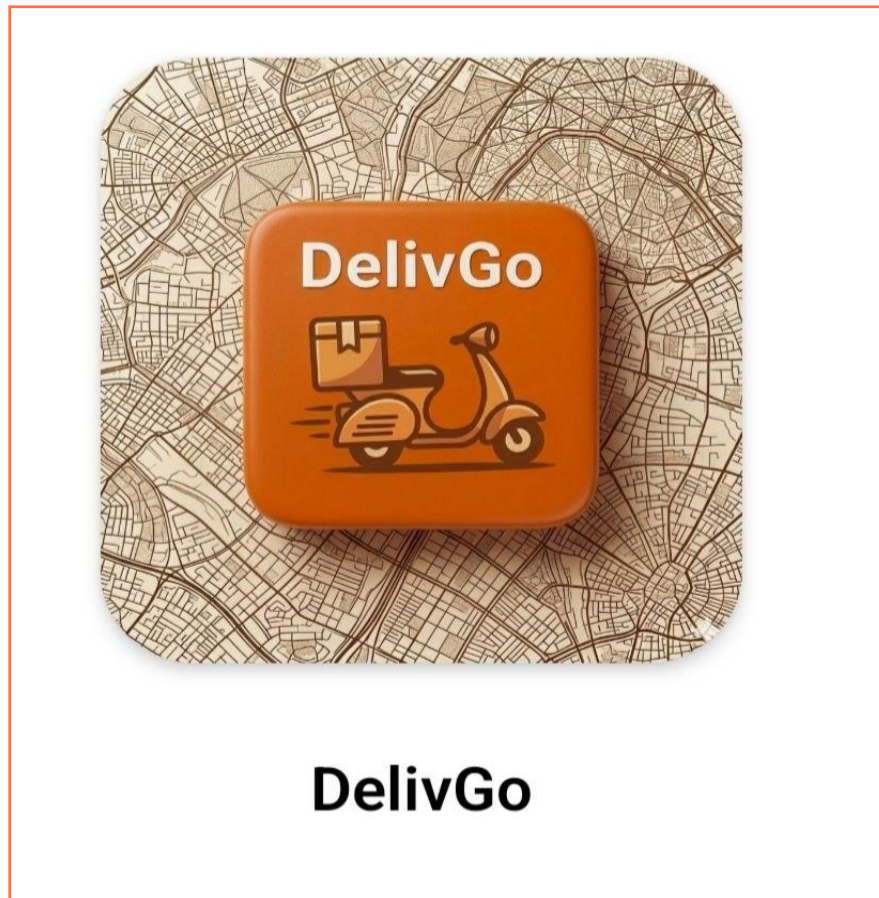
- **Node.js:** A JavaScript runtime environment used for developing the back-end part of applications. It enables efficient request processing, server-side operations, and supports asynchronous execution for high performance[14].
- **PostgreSQL:** An open-source relational database management system used to securely store and organize application data while providing stability, high performance, and support for complex database operations[15].
- **Express.js:** A lightweight and fast framework built on Node.js, used for developing web applications and servers. It simplifies API creation, request and response management, and route organization within the application[16].

2.2.3 Development Tools

- **VS Code:** Visual Studio Code is a lightweight and free development environment developed by Microsoft, used for writing and organizing source code. It provides features such as auto-completion, error detection, and debugging tools[17].
- **GitHub:** A cloud-based platform built on Git, used for managing software projects and storing different versions of source code while facilitating collaboration between developers[18].
- **Expo Go:** A development tool and mobile application used to test and run React Native applications directly on smartphones during the development phase, making it easier to preview and test modifications in real time[19].

3. Presentation of the System

3.1 App Logo



DelivGo

Fig. 3. 1: App Logo

3.2 Login page

The login page represents the main interface for accessing the application, allowing users to authenticate their accounts by entering their email address and password. The application also supports multilingual functionality, enabling users to choose between Arabic, French, and English languages in order to facilitate the use of the application for different users.

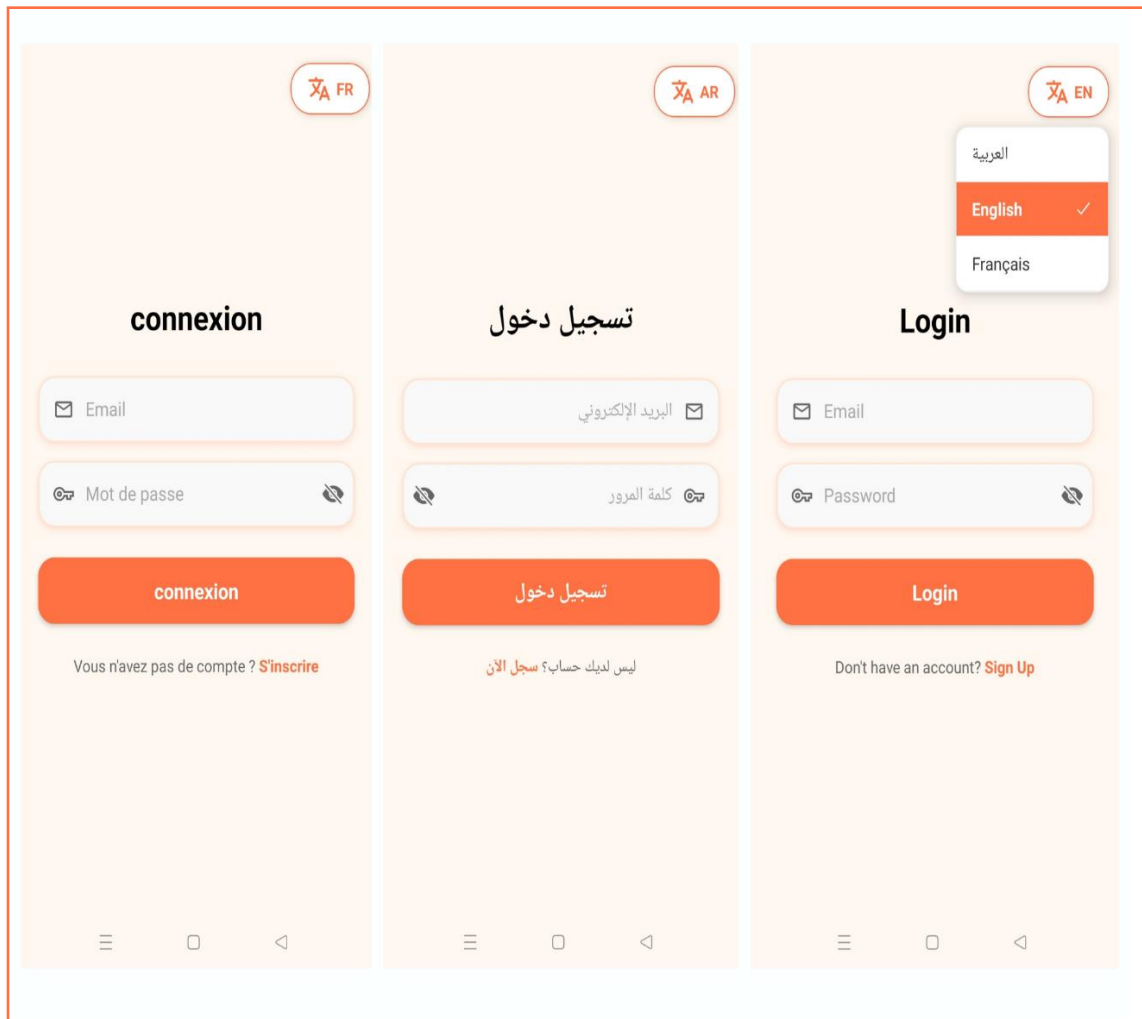


Fig. 3. 2: Login Page

3.3 Create Account page

The account creation interface represents the page dedicated to registering a new user in the application, where the user is allowed to enter their email address in order to create a new account. After that, the system sends a verification code to the user's email address to verify its validity before completing the registration process.

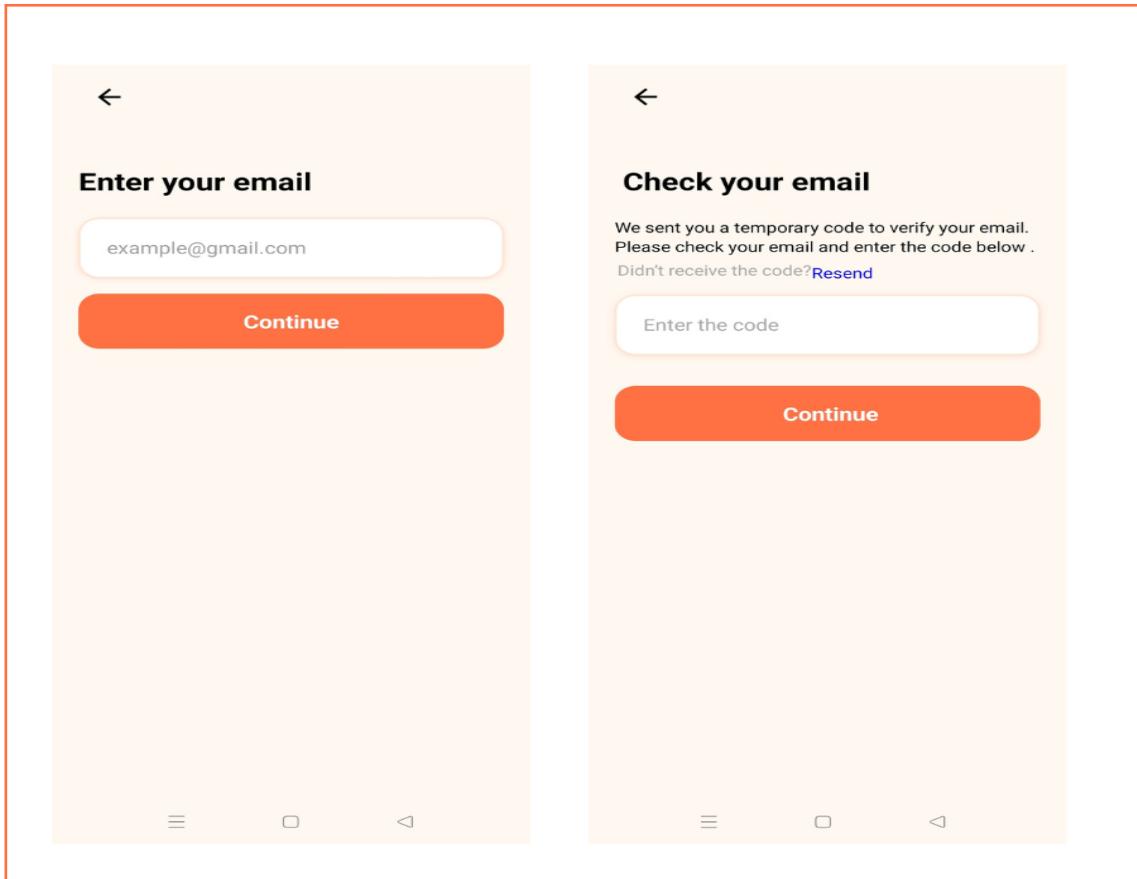


Fig. 3. 3: Verification page

After the email verification step, the account creation interface allows the user to enter their personal information. The user is also required to choose their role within the application, either as a client or a driver. Based on the selected role, the system displays the specific information and functionalities corresponding to each user role within the same interface.

Client: He enters:

- First name;
- Last name;
- Phone number;
- Role: Client;
- Password and confirm it.

The figure displays two versions of a 'Create an account' form side-by-side. Both forms have a back arrow at the top left and the title 'Create an account'. The left form has input fields for 'First name', 'Last name', and 'Phone number'. Below these is a 'Select Role:' section with two radio buttons: 'Client' (selected) and 'Driver'. At the bottom are 'Password' and 'Confirm password' fields with toggle icons. The right form has a 'Select Role:' section with 'Client' and 'Driver' radio buttons, where 'Driver' is selected. Below this are 'Password' and 'Confirm password' fields with toggle icons. A list of password requirements is shown: 'Make sure password contains: . At least 8 characters, . One uppercase, . One lowercase, . One symbol (+-@!/?#)'. An orange 'Continue' button is at the bottom.

Fig. 3. 4: Create account page-Client-

Driver: He enters:

- First name;
- Last name;
- Phone number;
- Role: Driver;
- Vehicle type selection ;
- The price (from-to) ;
- PDF ID Card insertion ;
- Password and confirm it.

The figure displays two mobile application screens for account creation, side-by-side.

Left Screen (Driver Account Creation):

- Back arrow icon at the top left.
- Title: **Create an account**
- Input fields: First name, Last name, Phone number.
- Section: **Select Role:**
 - ☐ Client
 - ☒ Driver
- Input field: Select vehicle... (dropdown menu)
- Input field: Price from 000,000 to 000,000

Right Screen (Client Account Creation):

- Back arrow icon at the top left.
- Title: **Create an account**
- Section: **PDF ID Card Insertion**
 - Download pdf button
- Input field: Password (with eye icon for visibility toggle)
- Input field: Confirm password (with eye icon for visibility toggle)
- Section: **Make sure password contains:**
 - . At least 8 characters
 - . One uppercase
 - . One lowercase
 - . One symbol (+-@!/?#)
- Continue button

Fig. 3. 5: Create account page-Driver-

3.3.1 Profile of client and driver page

The profile interface for both the client and the driver represents a dedicated space for managing account information and controlling some basic application settings. It allows the user to edit personal information, change the password, view the application description, and logout from the account.

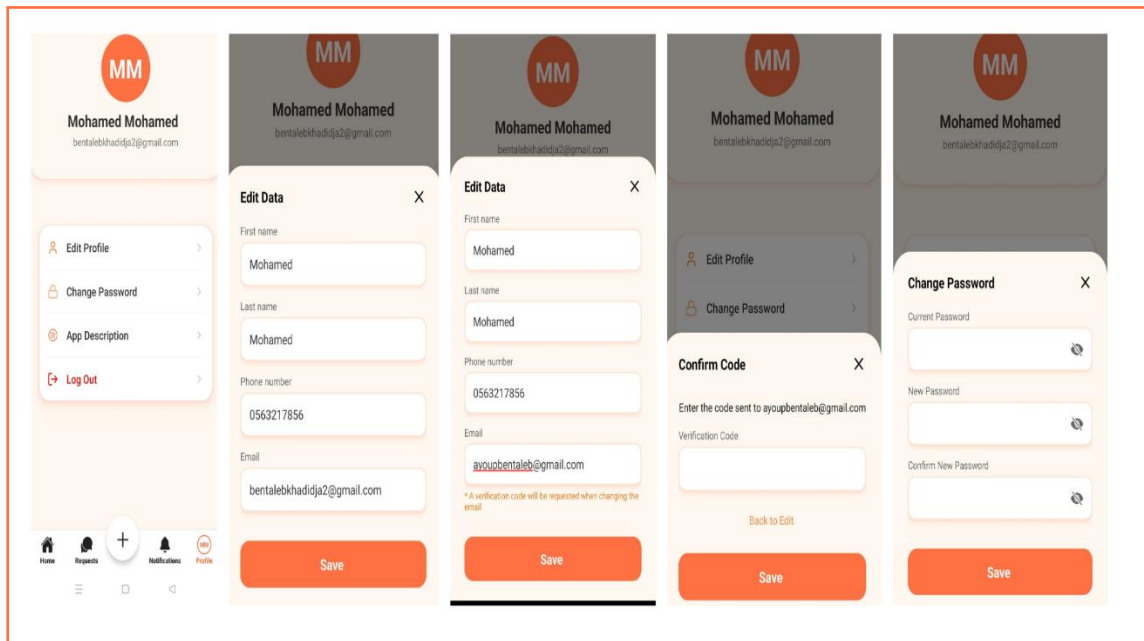


Fig. 3. 6: Profile of client

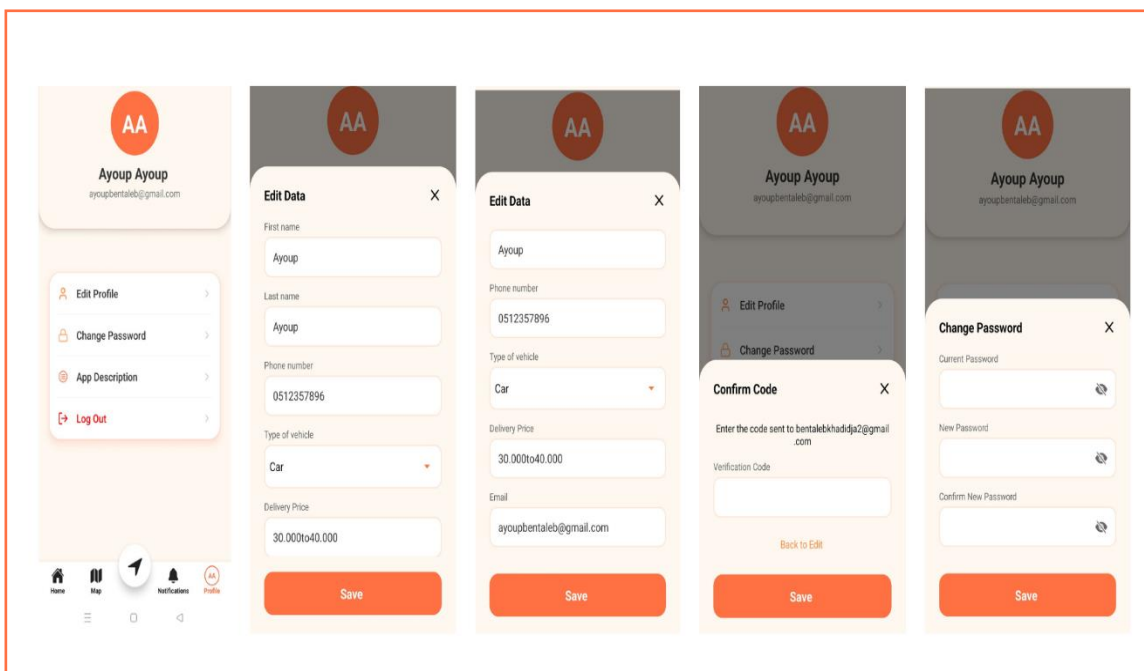


Fig. 3. 7: Profile of Driver

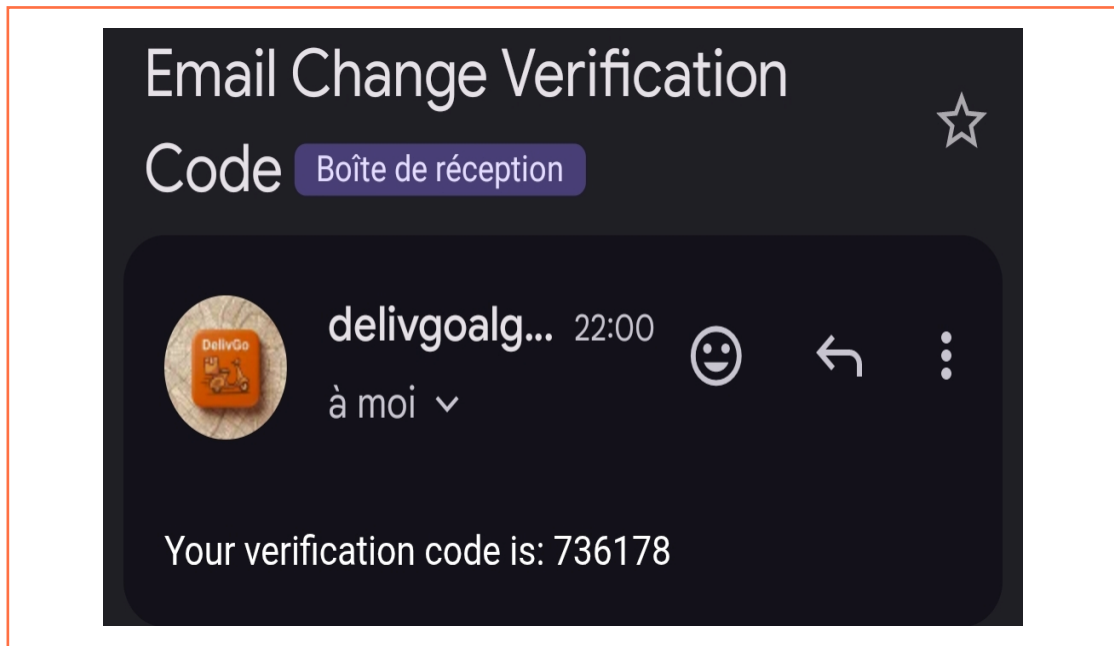


Fig. 3. 8: Verification Code

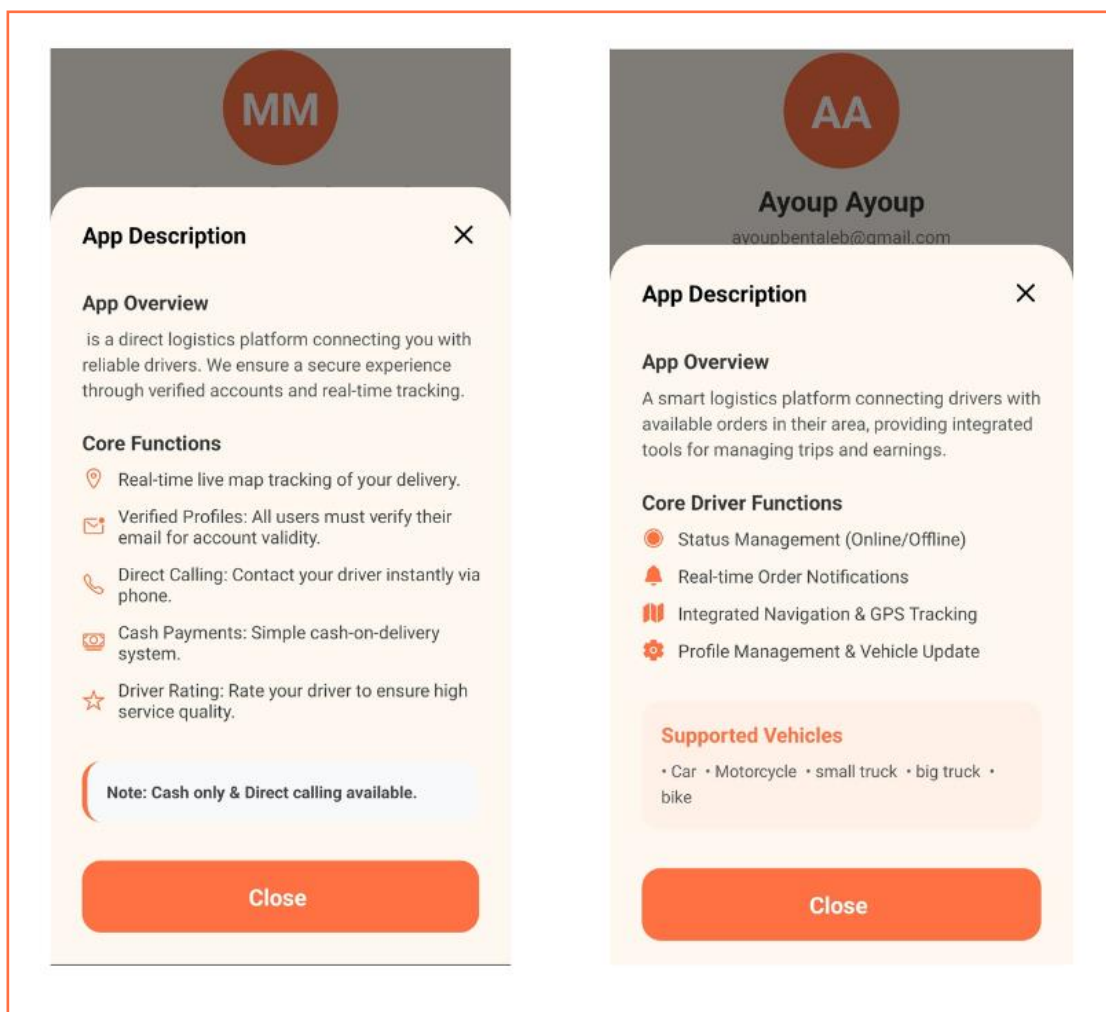


Fig. 3. 9: App Description

3.4 Create a new request page

This interface represents the page dedicated to sending delivery requests within the application, where it allows the user to enter various information related to the delivery process. This includes specifying the pickup location and delivery location, selecting the appropriate vehicle type for the delivery operation, and indicating whether the payment will be made on behalf of the recipient. The user can also enter a description of the product or parcel and add the recipient's email address.

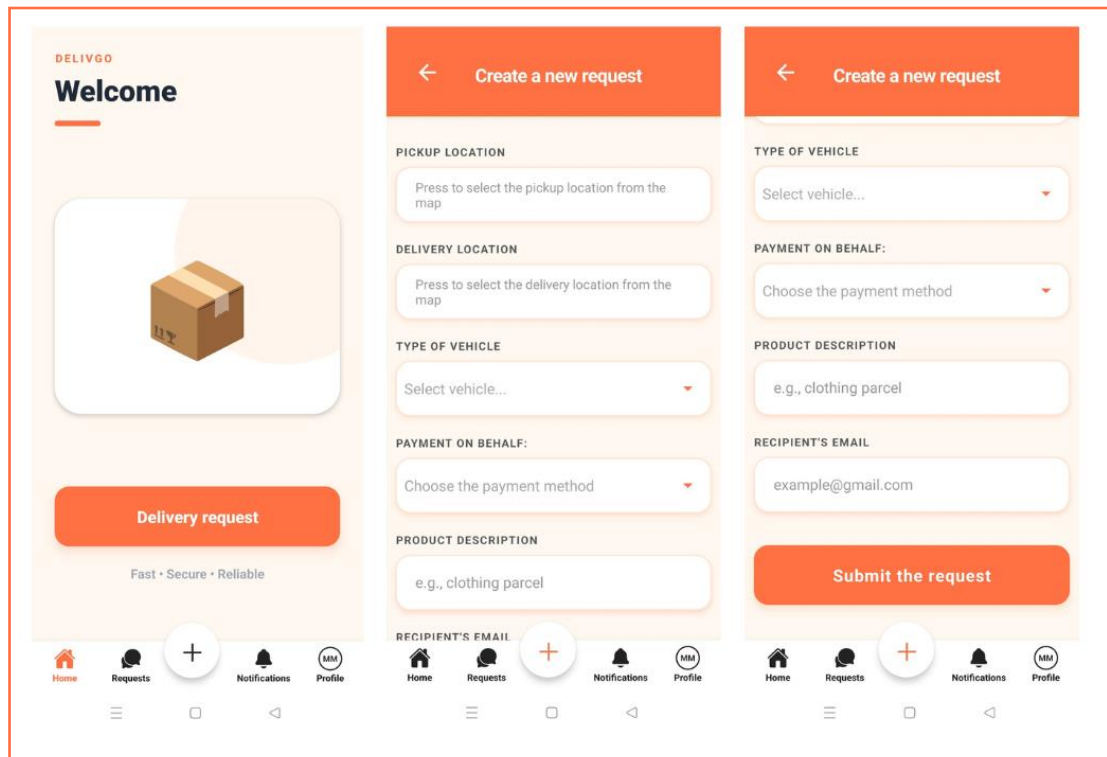


Fig. 3. 10: Create a new request

3.5 My Requests page

The My Requests interface represents the page dedicated to monitoring the different stages of delivery requests within the application, where it allows the client to view the status of requests. This interface displays request information after being accepted by the driver, including the order number, request status, product information, the driver responsible for the delivery process, and the payment amount. In return, the client can also confirm the start of the trip or cancel

it.

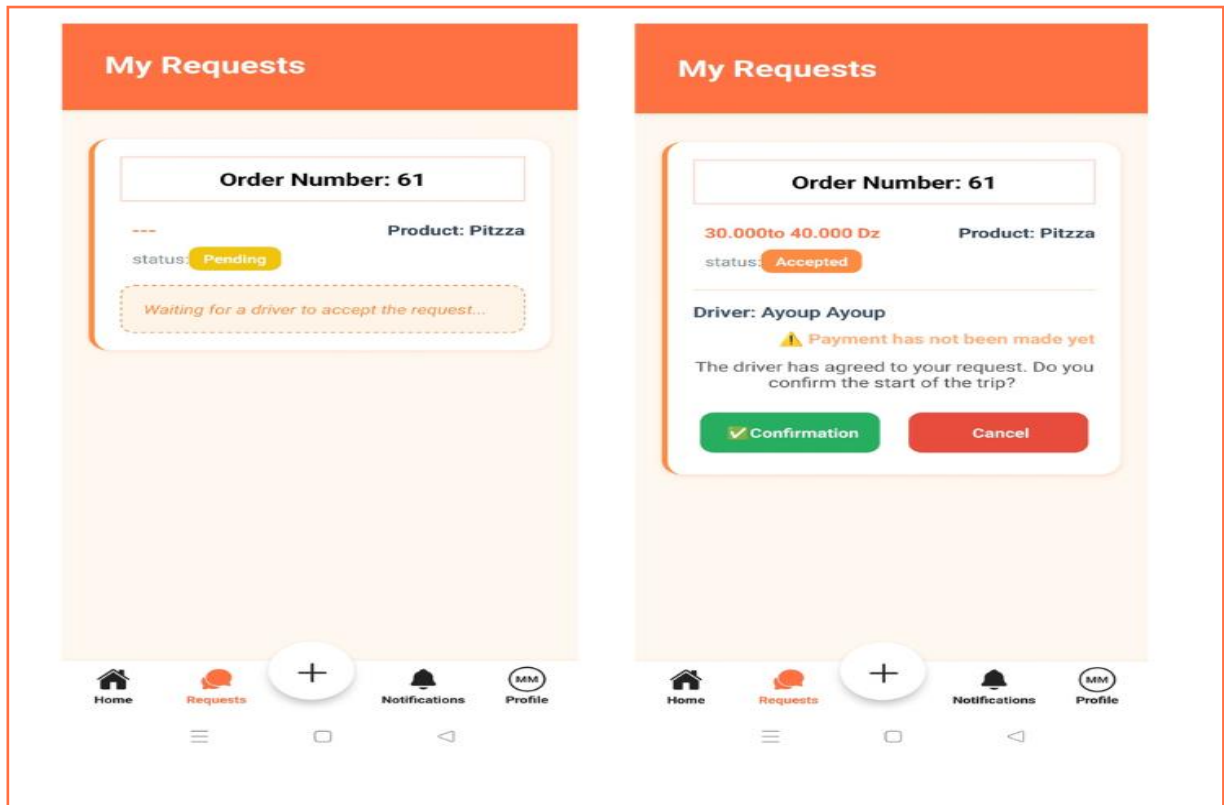


Fig. 3. 11: My Request-Client-

3.6 Home and Map page

This interface represents the main page for managing and monitoring delivery requests, where the driver can view available requests and accept or reject them. It also displays customer information, product details, and pickup and delivery locations, with the possibility of contacting the customer. In addition, the application provides another interface displaying a map that shows the shortest route between the pickup and delivery locations.

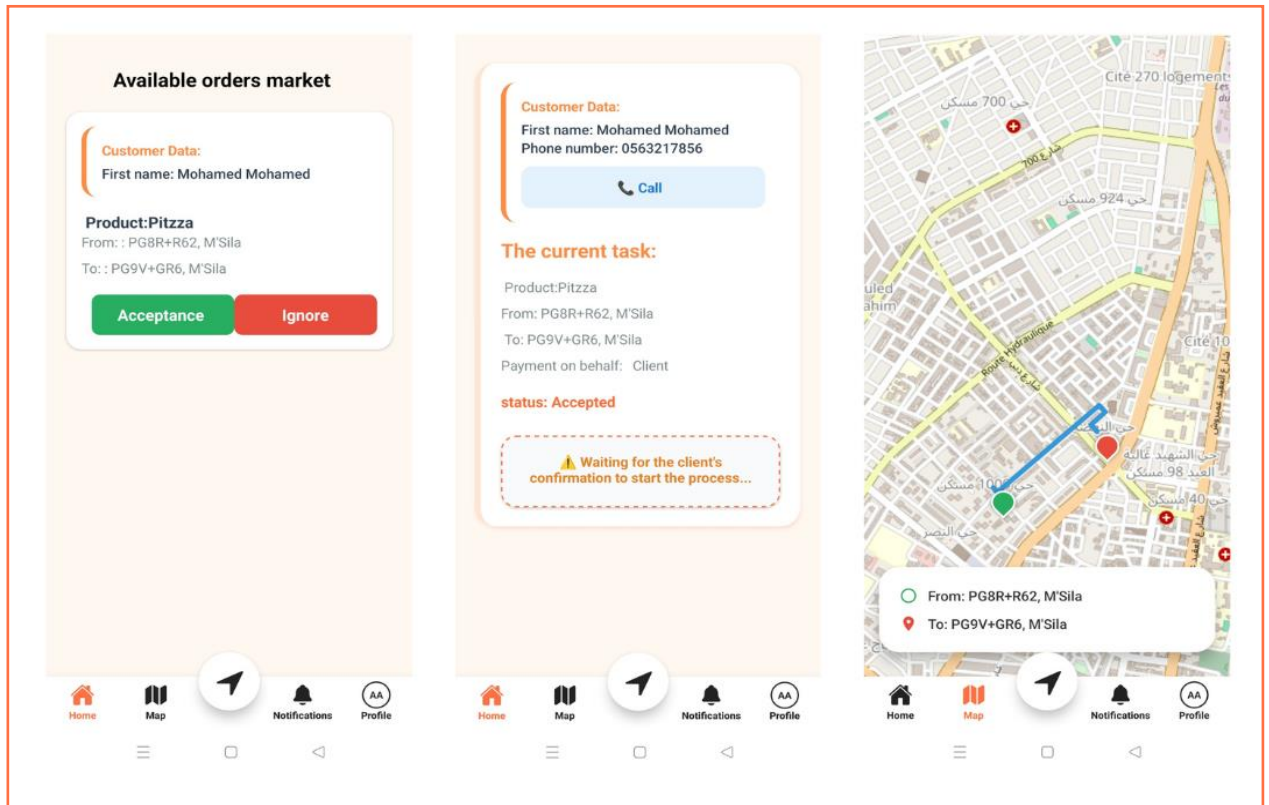


Fig. 3. 12: Home & Map -Driver-

In this interface, notifications related to previous operations are displayed within the application as well as through email.

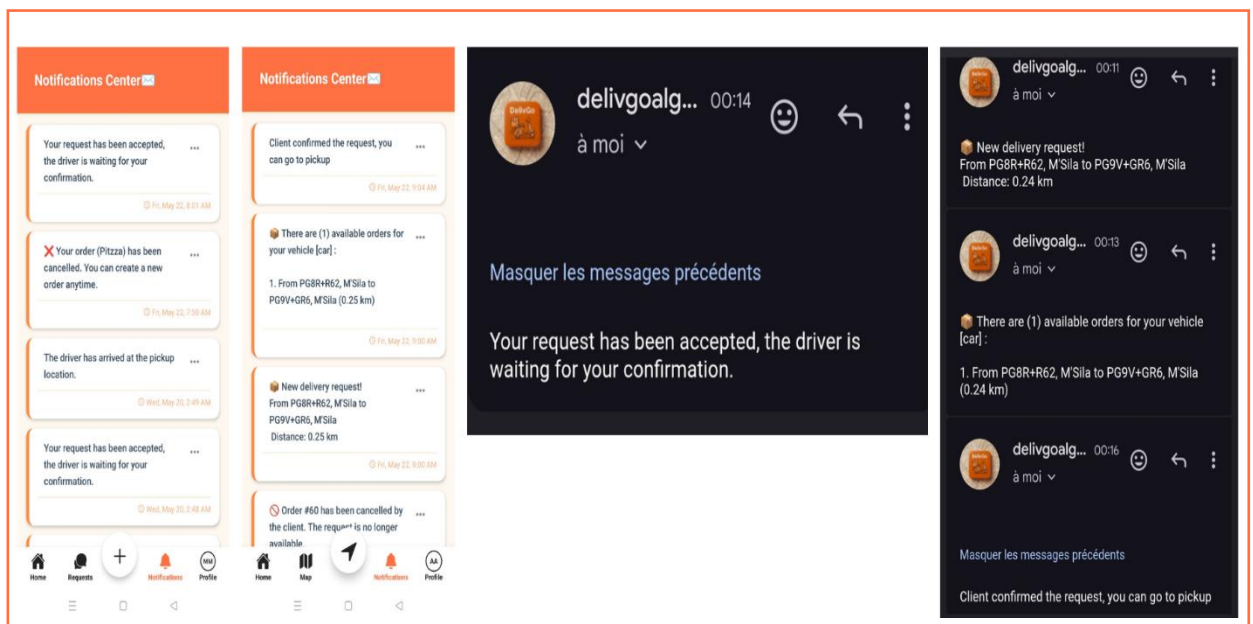


Fig. 3. 13: Notification Center

3.7 Order Delivery Lifecycle

3.7.1 Pickup Stage

Driver:

When the driver arrives at the pickup location, they press the “Arrived at Pickup Location” button, which sends a notification to the client informing them that the driver has arrived. After receiving the parcel and the payment amount, the driver presses the “Pickup Confirmation” and “Payment Confirmation” buttons. Then, the driver waits for the final confirmation from the client in order to continue the rest of the delivery process.

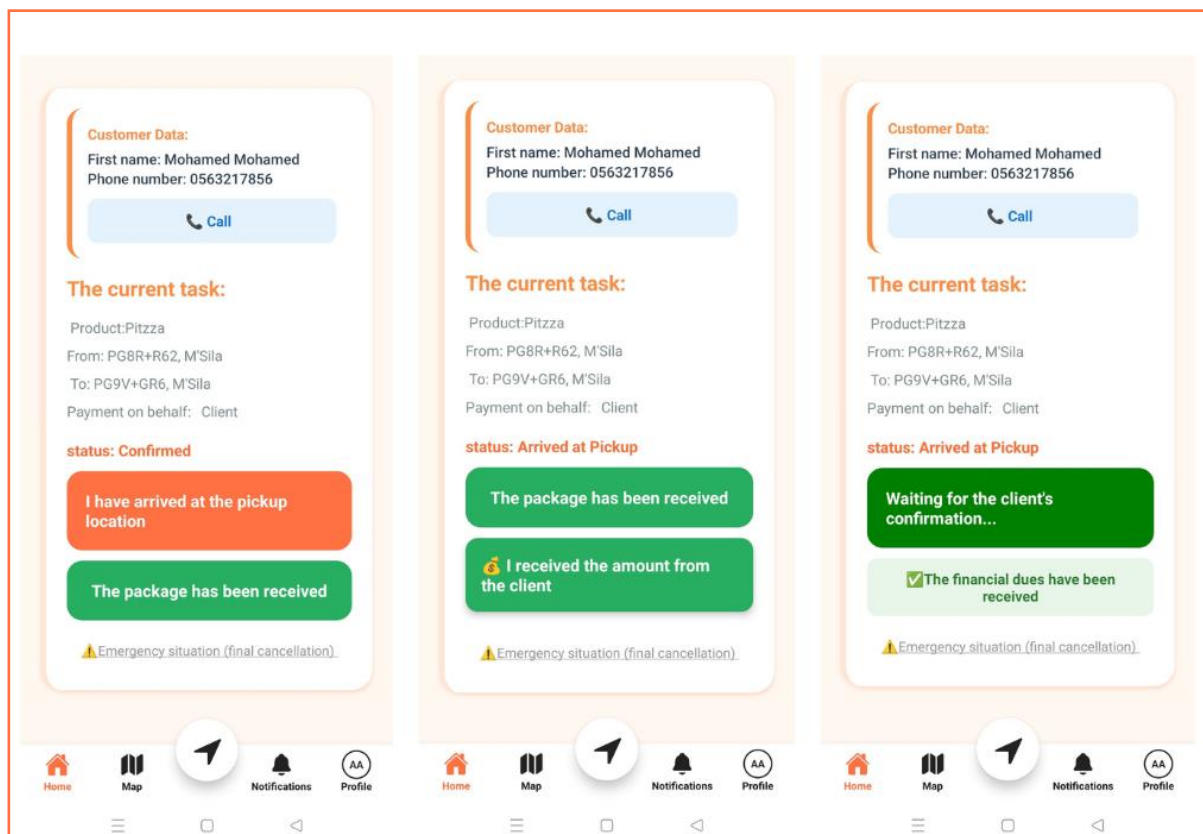


Fig. 3. 14: Pickup Stage-Driver-

Client :

A dedicated interface with the status Arrived at Pickup is displayed to the client, containing a button titled I handed the package to the driver, which the client presses after delivering the package to the driver. Once the pickup process is successfully completed, a special window appears allowing the client to evaluate their experience with the driver using a star rating system.

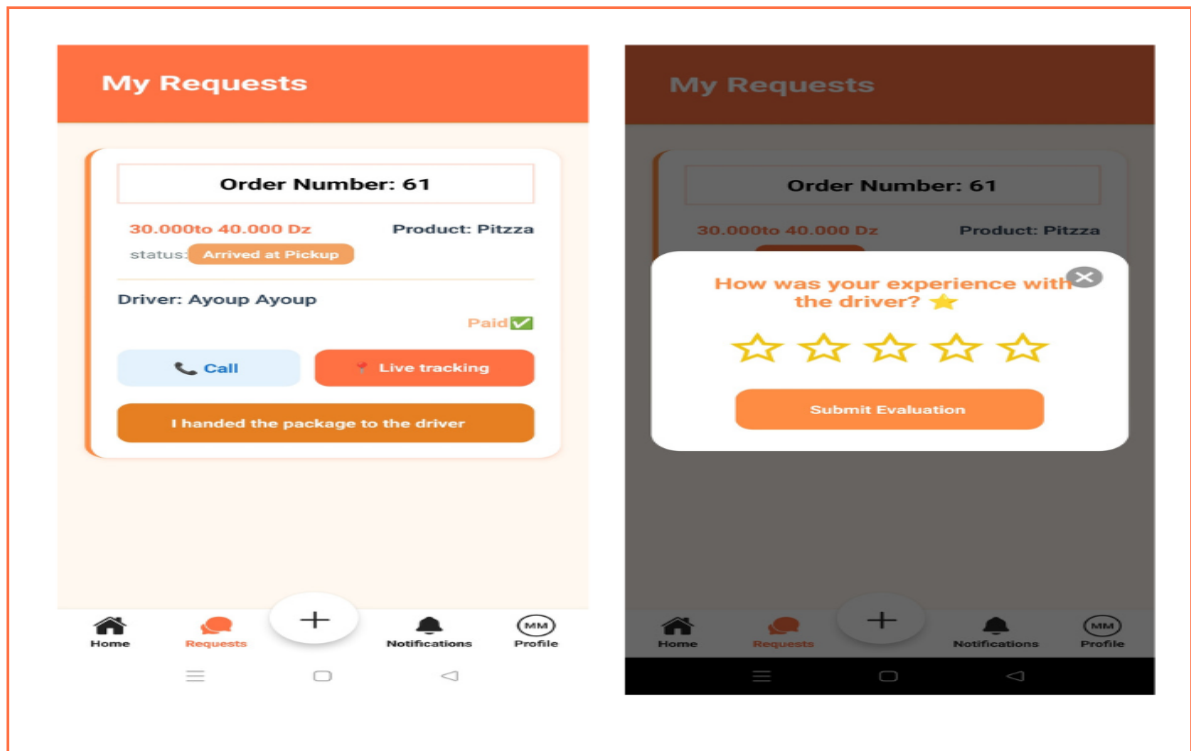


Fig. 3. 15:Pickup Stage-Client-

3.7.2 Delivery Stage

Driver :

When the driver arrives at the delivery location, they press the “Arrived at Delivery Location” button, which sends a notification to the recipient informing them that the driver has arrived. After handing over the package, the driver presses the “Delivery Confirmation” button, then waits for the final confirmation of receipt from the recipient.

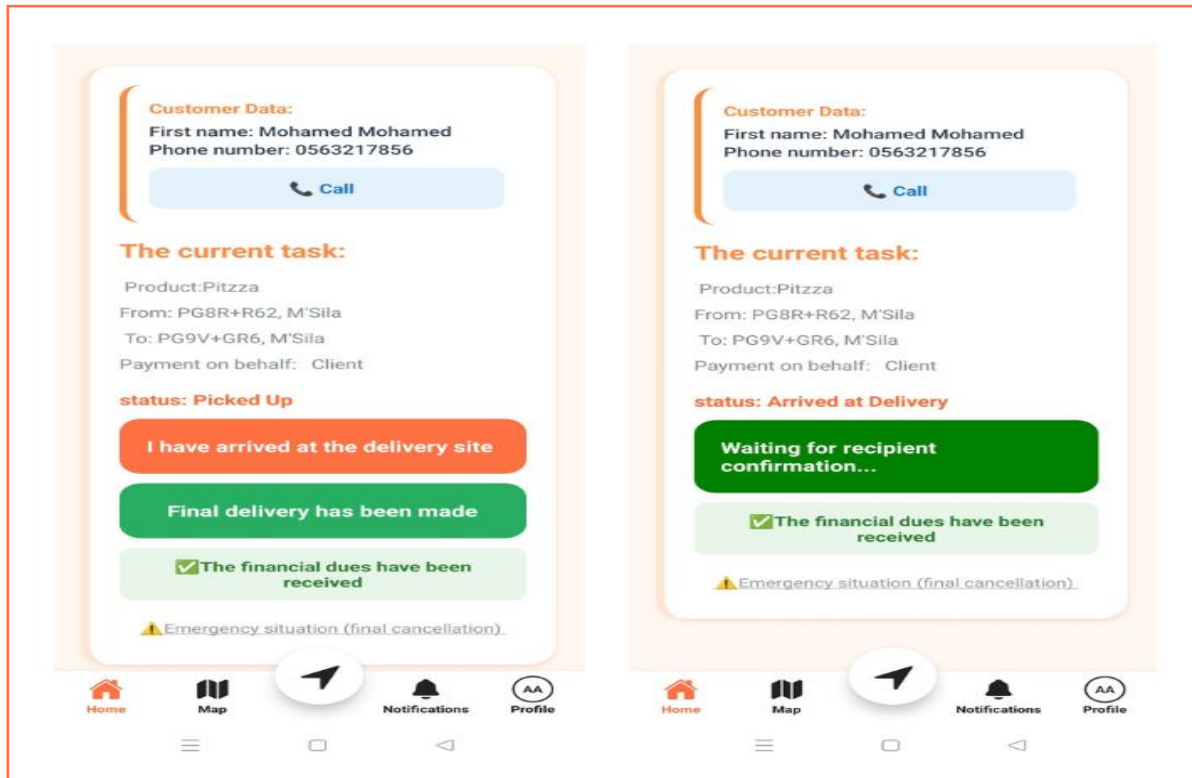


Fig. 3. 16:Delivery Stage-Driver-

Receiver :

The receiver is shown a dedicated interface with the status “Arrived at Delivery,” which includes a button labeled “I received the package” for confirming the delivery by the driver. The receiver presses this button after receiving the package. Once the receipt process is successfully completed, a dedicated window appears allowing the receiver to rate their experience with the driver using a star rating system.

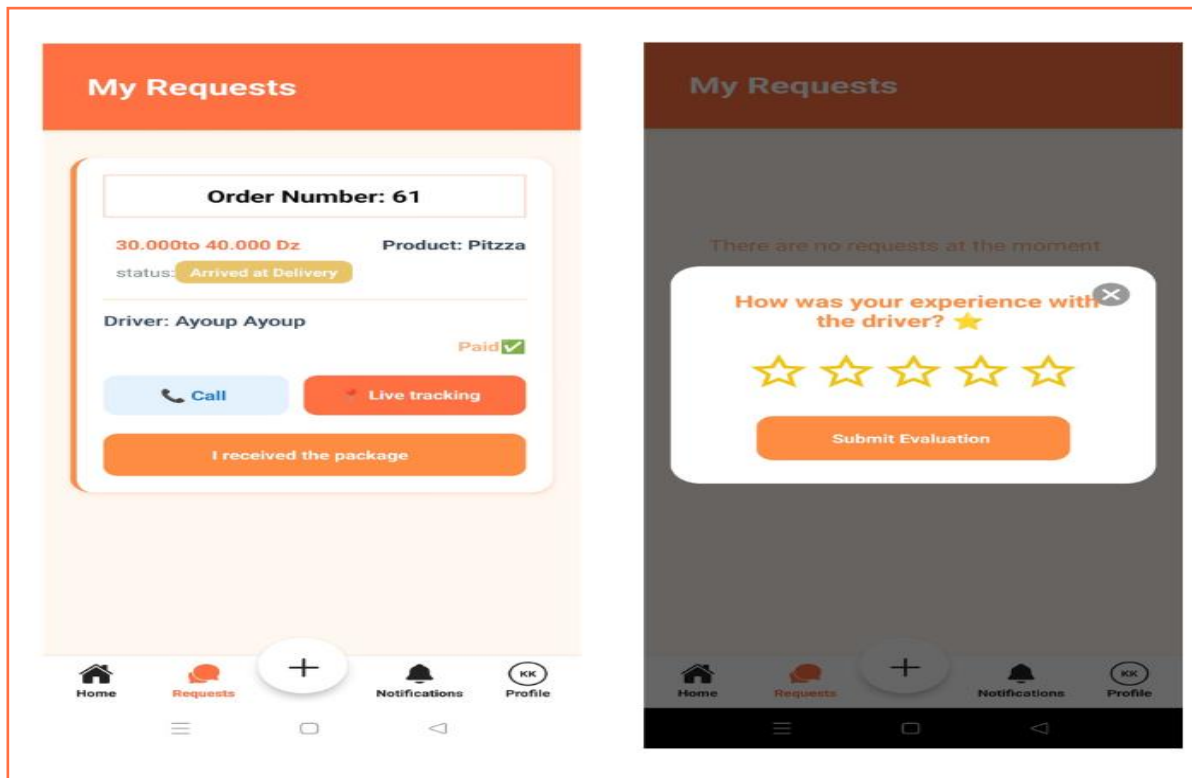


Fig. 3. 17:Delivery Stage-Receiver-

3.8 Tracking and Communication

These interfaces represent real-time driver tracking and communication with the driver.

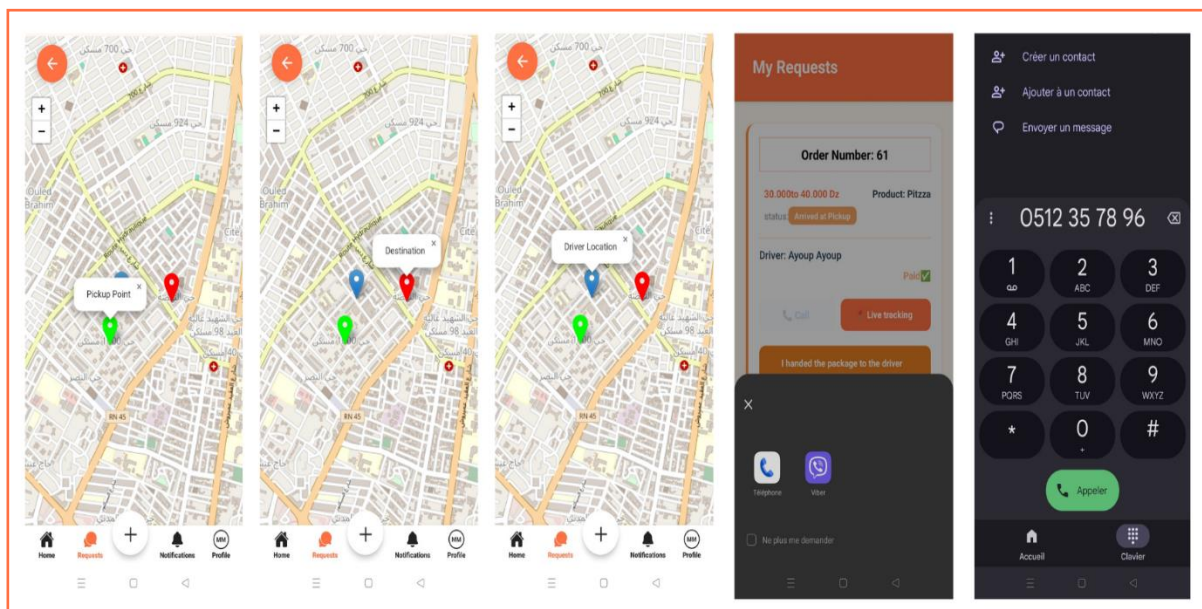


Fig. 3. 18: Tracking & call (Driver)

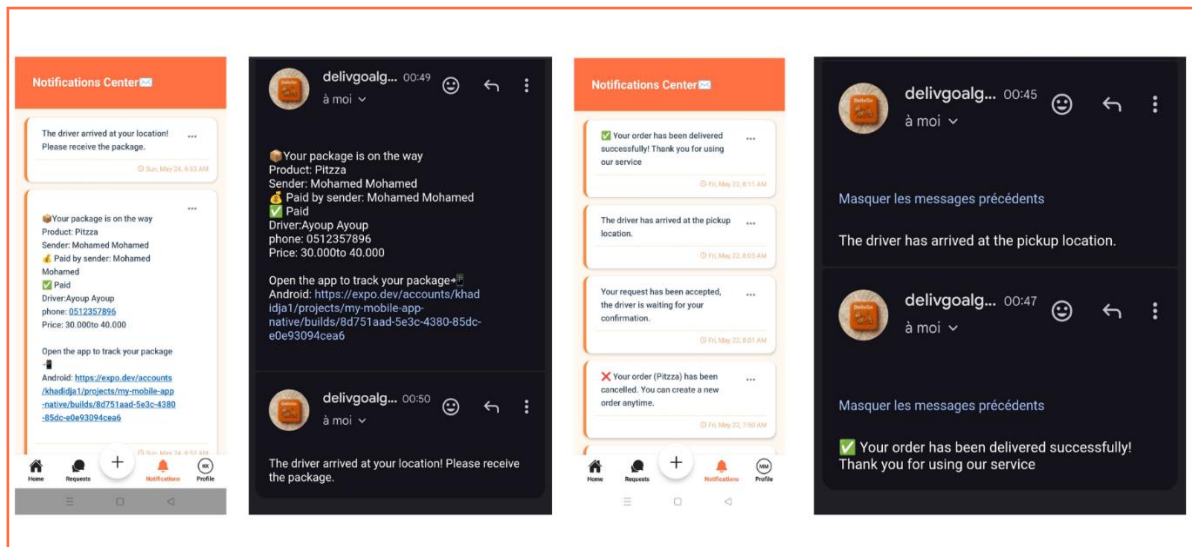


Fig. 3. 19: Notification (pickup & delivery) Stages

3.9 Cancel Request

After the driver accepts the request, he waits for the client's confirmation to start the process. If no confirmation is received, the driver cancels the request. The application then sends notifications to the driver that the request has been canceled, and informs the client that they can create a new request at any time.

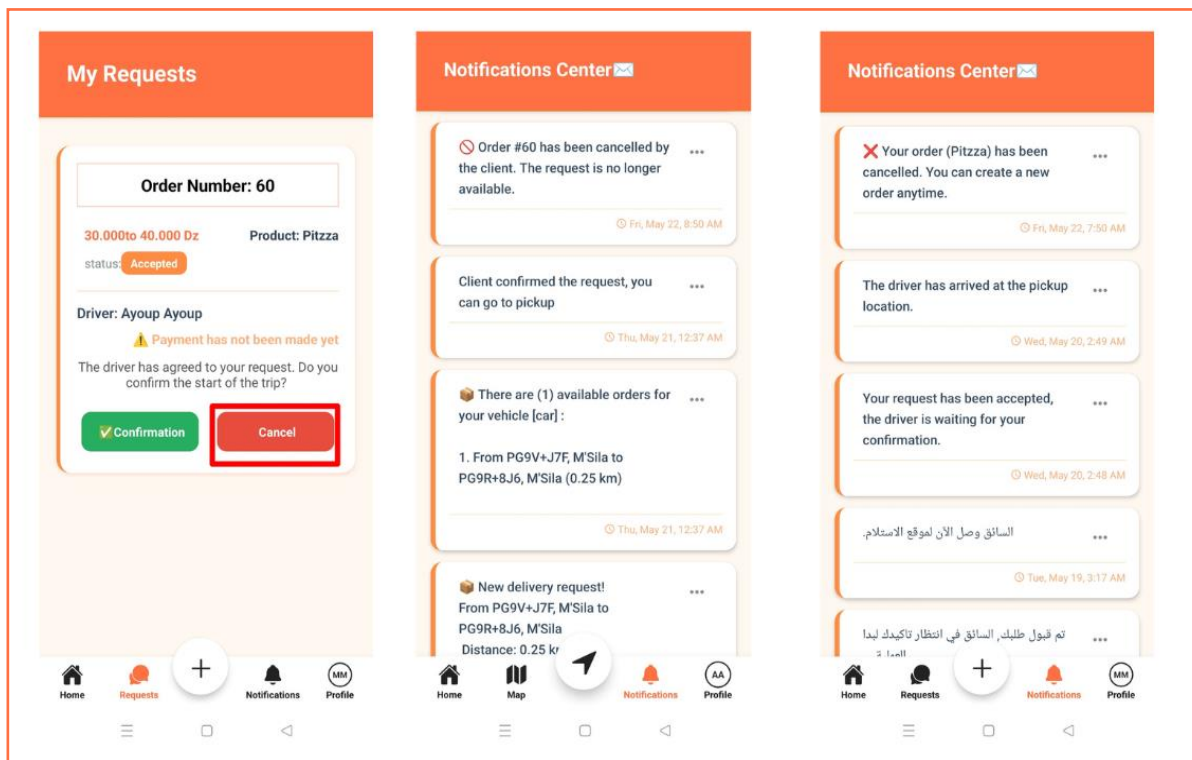


Fig. 3. 20: Cancel request(Client)

3.10 Emergency Situation

In case of an emergency during the delivery process, the driver can press the emergency button to cancel the operation.

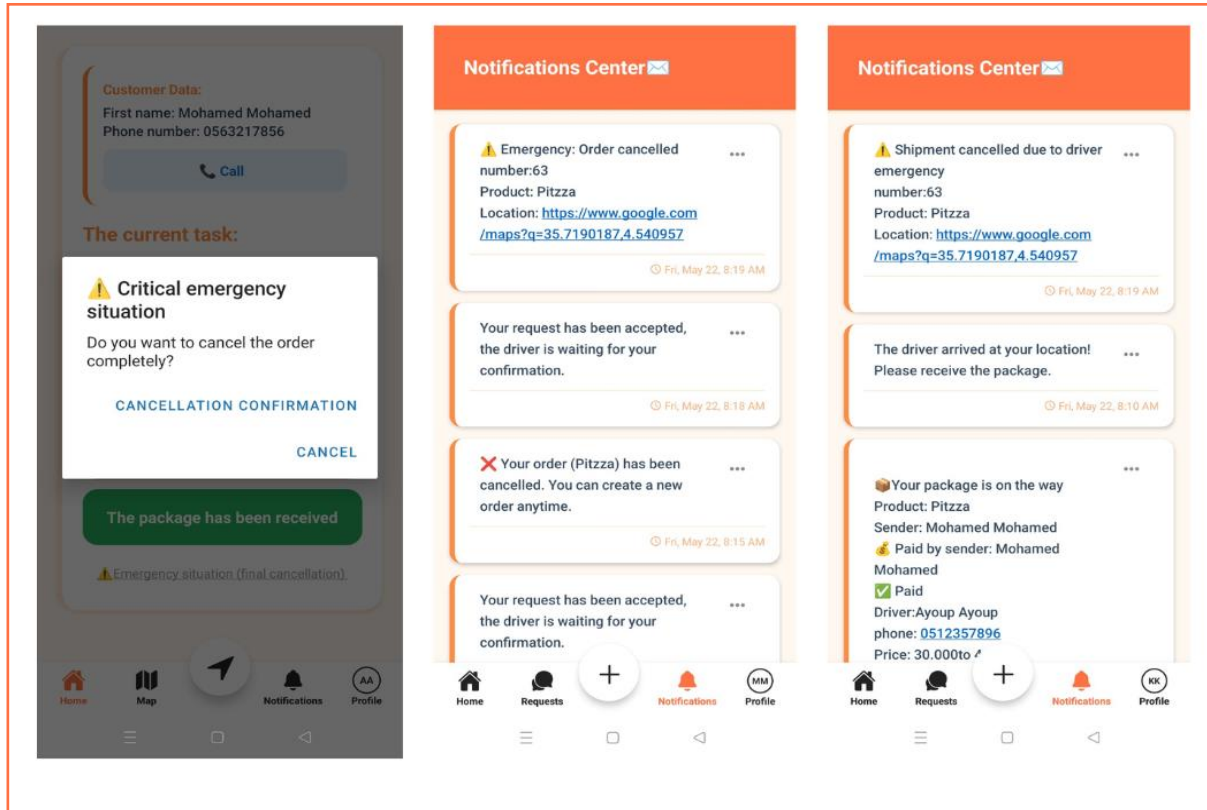


Fig. 3. 21:Emergency Situation

4.Application source code

This function represents:

- Updating the driver's status and retrieving the available requests assigned to him;
- Searching for and displaying the active request the driver is currently working on.

```
const fetchActiveDemand = async () => {
  try {
    const storedUser = await AsyncStorage.getItem("user_data");
    const user = JSON.parse(storedUser || "{}");
    let { status } = await Location.requestForegroundPermissionsAsync();
    let queryParams = `?driverId=${user.id}`;
    if (status === 'granted') {
      let loc = await Location.getLastKnownPositionAsync({});
      if (loc) {
        queryParams += `&lat=${loc.coords.latitude}&lng=${loc.coords.longitude}`;
      }
    }
    const res = await fetch(`${API_URL}/available-demands${queryParams}`);
    const data = await res.json();
    if (Array.isArray(data)) {
      const myJob = data.find((d) =>
        Number(d.delivery_id) === Number(user.id) && !(ultraTrim(d.status) === 'completed' || ultraTrim(d.status) === 'cancelled')
      );
      if (myJob) {
        setActiveDemand(myJob);
        setAvailableDemands([]);
      } else {

```

Fig. 3. 22: Fetch request

```
useEffect(() => {
  const startBackgroundTracking = async () => {
    try {
      const isProviderEnabled = await Location.hasServicesEnabledAsync();
      if (!isProviderEnabled) {
        try {
          await Location.enableNetworkProviderAsync();
        } catch (e) {
          console.log("User rejected location accuracy dialog");
        }
      }

      const { status: fgStatus } = await Location.requestForegroundPermissionsAsync();
      if (fgStatus !== 'granted') return;

      const { status: bgStatus } = await Location.requestBackgroundPermissionsAsync();
      if (bgStatus !== 'granted') {
        console.log("error");
        return;
      }

      const isRegistered = await TaskManager.isTaskRegisteredAsync(BACKGROUND_LOCATION_TASK);
      if (!isRegistered) {
        await Location.startLocationUpdatesAsync(BACKGROUND_LOCATION_TASK, {
          accuracy: Location.Accuracy.High,
          interval: 5000,
          distanceInterval: 1,
          showsBackgroundLocationIndicator: true,
          foregroundService: {
            notificationTitle: t('app_running'),
            notificationBody: t('tracking_location_msg'),
            notificationColor: "#FF7043"
          }
        });
      }
    } catch (error) {
      console.error(error);
    }
  };

  startBackgroundTracking();
}, []);
```

Fig. 3. 23: Update location the driver (Tracking)

```
const pool = new Pool({
  user: "postgres",
  host: "localhost",
  database: "dbase",
  password: process.env.DATE_BASE,
  port: 5432,
});
```

Fig. 3. 24: Database connection

```

316 // create request
317 app.post('/create-demand', async (req, res) => {
318   const { clientId, pickup, delivery, vehicle, product, recipientEmail, paymentBy } = req.body;
319   try {
320     const result = await pool.query(
321       `INSERT INTO demands (client_id, pickup_lat, pickup_lng, pickup_address, delivery_lat, delivery_lng, delivery_address, vehicle_type, product,
322        VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, 'pending', $11) RETURNING id,
323        [clientId, pickup.coords.latitude, pickup.coords.longitude, pickup.address, delivery.coords.latitude, delivery.coords.longitude, delivery.adc
324       );
325     const demandId = result.rows[0].id;
326
327     const drivers = await pool.query(
328       `SELECT email, id, latitude, longitude, average_delivery, delivery_type, lang
329       FROM delivery_dev
330       WHERE delivery_type = $1
331       AND NOT EXISTS (
332         SELECT 1
333         FROM demands
334         WHERE delivery_id = dev.id
335         AND status NOT IN ('completed', 'cancelled')
336       )
337       `, [vehicle]);
338
339     const nearbyVIPs = [];
340     const nearbyNormals = [];
341
342     drivers.rows.forEach(driver => {
343       const distance = calculateDistance(pickup.coords.latitude, pickup.coords.longitude, driver.latitude, driver.longitude);
344
345       if (distance <= 3) {
346
347         const payload = {
348           pickup: pickup.address,
349           delivery: delivery.address,
350           distance: distance.toFixed(2)
351         };
352
353         if (driver.average_delivery > 5) nearbyVIPs.push({ ...driver, payload });
354         else nearbyNormals.push({ ...driver, payload });
355
356       }
357     });
358
359     nearbyVIPs.forEach(d =>
360       sendDualNotification(
361         d.id,
362         d.email,
363         "delivery",
364         "new_demand",
365         d.payload
366       )
367     );
368     else {
369       nearbyNormals.forEach(d =>
370         sendDualNotification(
371           d.id,
372           d.email,
373           "delivery",
374           "new_demand",
375           d.payload
376         )
377       );
378     }
379
380     res.json({ success: true, demandId });
381   } catch (err) { res.status(500).json({ error: er
382   ;

```

Fig. 3. 25: Create a request and search for a driver

5. Conclusion

Through this chapter, the practical aspect of the project was presented by introducing the various tools and technologies used in the development of the application, as well as presenting the main implemented interfaces and core functionalities it provides, such as sending delivery requests, geolocation tracking, notifications, and delivery management. The implementation phase made it possible to transform the theoretical designs and models prepared in the previous chapter into a real and usable application that achieves the main objectives of the project and provides an effective and seamless user experience.

General Conclusion

The delivery services sector has witnessed rapid development thanks to the significant advancement in digital technologies and geolocation systems, which has led to the emergence of smart solutions aimed at improving the speed and accuracy of services. In this context, a smart delivery application based on real-time geolocation technologies was designed and developed in order to improve and simplify delivery operations between users. The proposed system aimed to provide an efficient solution that connects the client, driver, and receiver through an interactive platform capable of ensuring fast delivery, accurate tracking, and improved service quality.

This work started with a theoretical study covering delivery concepts and geolocation tracking technologies, followed by the system analysis and design phase using UML diagrams to clearly and systematically define the different functionalities and relationships between the system components. After that, the practical aspect was implemented through the development of a functional application using a set of modern technologies. The application provides several important features, including real-time tracking, automatic driver selection, shortest-path calculation, instant notifications, and flexible delivery management, which contributed to improving user experience and enhancing service efficiency. However, some limitations still remain, such as the need to strengthen security aspects and integrate electronic payment methods.

As future work, artificial intelligence techniques can be integrated to improve driver recommendation and route prediction, in addition to supporting online payment services and extending the application features to include more advanced and efficient functionalities.

References

- [1] A. Ghiani, G. Laporte, and R. Musmanno, “Introduction to logistics systems management,” *Springer (open previews via Scholar)*. [Online]. Available: <https://scholar.google.com/scholar?q=real+time+tracking+delivery+system+IoT+logistics> [Accessed: january 23, 2026].
- [2] P. Toth and D. Vigo, “Vehicle routing: Problems, methods, and applications,” *ResearchGate / Springer preview*, 2014. [Online]. Available: <https://scholar.google.com/scholar?q=vehicle+routing+problem+delivery+optimization+manual+system> [Accessed: january 24, 2026].
- [3] Geotab, “GPS Tracking,” Geotab. [Online]. Available: <https://www.geotab.com/>. [Accessed: February 15, 2026]
- [4] S. He and S.-H. G. Chan, “Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons,” *ResearchGate / Scholar results*. [Online]. Available: <https://scholar.google.com/scholar?q=WiFi+fingerprinting+indoor+positioning+survey> [Accessed: February 23, 2026].
- [5] P. Del Peral-Rosado, R. Raulefs, J. A. López-Salcedo, and G. Seco-Granados, “A Survey of Indoor Positioning Systems: Technologies, Techniques and Applications,” *Preprint (2017), Universitat Autònoma de Barcelona*. [Online]. Available: https://spcomnav.uab.es/docs/journals/DelPeral_Survey_preprint_2017.pdf [Accessed: February 23, 2026].
- [6] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, “BLE Beacons for Internet of Things Applications,” *IEEE IoT Journal*. [Online]. Available: <https://scholar.google.com/scholar?q=BLE+beacon+indoor+positioning+survey> [Accessed: February 23, 2026].
- [7] Object Management Group (OMG), “Unified Modeling Language (UML),” *OMG Standard*. [Online]. Available: <https://www.omg.org/spec/UML/>.

- [Accessed: March 25, 2026]
- [8] Meta Platforms, “React Native Documentation,” Meta.
[Online]. Available: <https://reactnative.dev/docs/getting-started>
[Accessed: May 23, 2026].
- [9] OpenStreetMap Foundation, “OpenStreetMap Wiki,” OSM.
[Online]. Available: https://wiki.openstreetmap.org/wiki/Main_Page
[Accessed: May 23, 2026].
- [10] Heigit – Heidelberg Institute for Geoinformation Technology, “OpenRouteService API Documentation.”
[Online]. Available: <https://openrouteservice.org/dev/#/api-docs>
[Accessed: May 23, 2026].
- [11] MDN Web Docs, “JavaScript Guide,” Mozilla.
[Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
[Accessed: May 23, 2026].
- [12] Microsoft, “TypeScript Handbook,” Microsoft.
[Online]. Available: <https://www.typescriptlang.org/docs/>
[Accessed: May 23, 2026].
- [13] i18next, “Internationalization Framework Documentation,”
[Online]. Available: <https://www.i18next.com/>
[Accessed: May 23, 2026].
- [14] OpenJS Foundation, “Node.js Documentation,” Node.js.
[Online]. Available: <https://nodejs.org/en/docs>
[Accessed: May 23, 2026].
- [15] PostgreSQL Global Development Group, “PostgreSQL Documentation,” PostgreSQL.
[Online]. Available: <https://www.postgresql.org/docs/>
[Accessed: May 23, 2026].

- [16] OpenJS Foundation, “Express Documentation,” Express.js.
[Online]. Available: <https://expressjs.com/>
[Accessed: May 23, 2026].

- [17] Microsoft, “Visual Studio Code Documentation,” Microsoft.[Online]. Available:
<https://code.visualstudio.com/docs>
[Accessed: May 16, 2026].

- [18] GitHub, “About GitHub,” GitHub.
[Online]. Available: <https://docs.github.com/en/get-started/quickstart/hello-world>
[Accessed: May 16, 2026].

- [19] Expo, “Expo Documentation,” Expo.
[Online]. Available: <https://docs.expo.dev/>
[Accessed: May 23, 2026].