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

AISSAT AYMEN

RIGHI ZAID

Title of the thesis

Design &Implementation of a Full-Stack Mobile Vehicle Insurance Application with AI-Powered Biometric Identity Verification

---

Under the supervision of

AKHROUF SAMIR

Composition of the jury

BOUGUERR Abdelbaki	University of Msila	President
AKHROUF Samir	University of Msila	Reporter
BOUKHARI Nawel	University of Msila	Examiner

06, 2026

Abstract

This work aims to design and develop a comprehensive digital vehicle insurance application that simplifies insurance operations and improves the user experience, leveraging modern web and mobile technologies. The proposed system uses a multi-layer architecture that includes a Flutter mobile app, a React and TypeScript administrative dashboard, a Node.js and Express.js backend, and a facial verification microservice built with Flask and DeepFace. The application offers user registration, vehicle registration, insurance policy subscription, claim submission, document uploading, online payment processing, and real-time status tracking. The administrative dashboard allows experts and administrators to manage policies, review claims, and generate assessment reports. To improve security, the platform integrates JWT-based authentication and biometric face verification. MongoDB serves as the primary database, while Socket.io provides real-time notifications and communication between system components. Functional testing confirmed the successful operation of all modules, including authentication, policy management, claims processing, payment integration, notification services, and identity verification. These results demonstrate the effectiveness of the proposed solution and its potential as a modern digital platform for managing vehicle insurance.

Keywords: Vehicle Insurance, Mobile Application, Flutter, Node.js, MongoDB, DeepFace, Identity Verification, Claims Management, Online Payment.

الملخص

يهدف هذا العمل إلى تصميم وتطوير تطبيق رقمي شامل لتأمين المركبات، يُبسّط عمليات التأمين ويُحسّن تجربة المستخدم، بالاستفادة من أحدث تقنيات الويب والهواتف المحمولة. يستخدم النظام المقترح بنية متعددة الطبقات تتضمن تطبيقًا للهواتف المحمولة مبنيًا بتقنية Flutter، ولوحة تحكم إدارية مبنية بتقنية React وTypeScript، وخادمًا خلفيًا مبنيًا بتقنية Node.js وExpress.js، وخدمة مصغّرة للتحقق من الوجه باستخدام Flask وDeepFace. يوفر التطبيق خدمات تسجيل المستخدمين والمركبات، والاشتراك في وثائق التأمين، وتقديم المطالبات، وتحميل المستندات، ومعالجة المدفوعات عبر الإنترنت، وتتبع الحالة في الوقت الفعلي. تُمكّن لوحة التحكم الإدارية الخبراء والمسؤولين من إدارة الوثائق، ومراجعة المطالبات، وإنشاء تقارير التقييم. ولتعزيز الأمان، تدمج المنصة المصادقة القائمة على JWT والتحقق البيومتري من الوجه. تُستخدم MongoDB كقاعدة البيانات الرئيسية، بينما توفر Socket.io الإشعارات والتواصل في الوقت الفعلي بين مكونات النظام. أكدت الاختبارات الوظيفية التشغيل الناجح لجميع الوحدات، بما في ذلك المصادقة، وإدارة الوثائق، ومعالجة المطالبات، وتكامل الدفع، وخدمات الإشعارات، والتحقق من الهوية. تُظهر هذه النتائج فعالية الحل المقترح وإمكاناته كمنصة رقمية حديثة لإدارة تأمين المركبات.

الكلمات المفتاحية: تأمين المركبات، تطبيق محمول، Flutter، Node.js، MongoDB، DeepFace، التحقق من الهوية، التصريح بالحوادث، الدفع الإلكتروني.

Résumé

Ce travail vise à concevoir et développer une application numérique complète d'assurance automobile, simplifiant les opérations et améliorant l'expérience utilisateur grâce aux technologies web et mobiles modernes. Le système proposé utilise une architecture multicouche comprenant une application mobile Flutter, un tableau de bord d'administration React et TypeScript, un backend Node.js et Express.js, et un microservice de vérification faciale développé avec Flask et DeepFace. L'application offre l'inscription des utilisateurs et des véhicules, la souscription à des polices d'assurance, la déclaration de sinistre, le téléchargement de documents, le traitement des paiements en ligne et le suivi en temps réel. Le tableau de bord d'administration permet aux experts et aux administrateurs de gérer les polices, d'examiner les sinistres et de générer des rapports d'évaluation. Pour renforcer la sécurité, la plateforme intègre l'authentification JWT et la vérification biométrique faciale. MongoDB sert de base de données principale, tandis que Socket.io assure les notifications en temps réel et la communication entre les composants du système. Les tests fonctionnels ont confirmé le bon fonctionnement de tous les modules, notamment l'authentification, la gestion des polices, le traitement des sinistres, l'intégration des paiements, les services de notification et la vérification d'identité. Ces résultats démontrent l'efficacité de la solution proposée et son potentiel en tant que plateforme numérique moderne de gestion de l'assurance automobile.

Mots-clés: Assurance automobile, application mobile, Flutter, Node.js, MongoDB, DeepFace, vérification d’identité, gestion des sinistres, paiement en ligne.

Dedications

" To those whose names I proudly bear, to those under whose care I grew up...

To the wellspring of boundless love and generosity... my dear mother, whose prayers were the secret to my success, and whose words were a balm to my path.

To the guiding light of my life and my support... my beloved father, who never hesitated to offer me support or advice, and from whom I learned patience and perseverance.

To my brothers and sisters, my companions on this journey, who shared with me every moment of joy and hardship.

To all those who illuminated my mind with their knowledge, my esteemed teachers, who generously shared their guidance and wisdom.

I dedicate this humble work to you, the fruit of your labor and sleepless nights, in recognition of your kindness and as an expression of my deepest gratitude."

Righi Zaid

" I dedicate this work to the dearest people in my heart, my dear parents, in appreciation of the love, care, and continuous support they have given me throughout my academic journey. They have been my greatest support and help at every step.

To my siblings and family members, who gave me encouragement and confidence, and shared with me moments of hardship and hope.

To my esteemed professors, who contributed to my academic and intellectual development, and whose guidance and advice had a significant impact on the completion of this work.

To my friends and colleagues, who accompanied me throughout my years of study and shared its experiences and challenges with me.

And to everyone who contributed, near or far, to supporting and assisting me, I dedicate this humble work as a token of thanks and gratitude..."

AISSAT Aymen

Acknowledgments

Acknowledgement First and foremost, we would like to express our sincere gratitude to Allah for granting us the strength, patience, and perseverance to complete this work. We are deeply grateful to our thesis supervisor Prof. Akhrouf Samir, for his valuable guidance, continuous support, insightful advice, and encouragement throughout the development of this project. His expertise and dedication played a crucial role in the successful completion of this work. We would also like to extend our heartfelt thanks to all our teachers and academic staff for the knowledge and skills they have shared with us during our years of study. Special thanks go to our families and friends for their unwavering support, understanding, and motivation during the challenging moments of this project. Finally, we would like to thank our institution for providing us with the opportunity to work on this project and strengthen our teamwork, communication, and problem-solving skills. We also express our appreciation to all team members for their cooperation, commitment, and team spirit throughout this journey. Thank you to everyone who contributed, directly or indirectly, to the realization of this work.

Contents

List of figures

List of tables

Introduction

The insurance sector has experienced significant digital transformation in recent years, driven by the rapid growth of mobile technologies and online services. Users today expect fast, secure, and accessible insurance solutions that allow them to manage their policies, submit claims, and complete administrative procedures directly from their smartphones, without the need to visit physical agencies or rely on paper-based processes.

However, despite this progress, many insurance systems still suffer from notable limitations. Insurance operations in algeria continue to depend on manual procedures, physical documentation, and in-person visits, which slow down processing times and negatively affect the overall customer experience. This is particularly evident in vehicle insurance, where customers frequently require quick access to claims management, payment operations, and policy tracking.

This work addresses these challenges by focusing on three main research aspects: the digital transformation of the insurance sector and its integration with mobile services, the need for secure and reliable user authentication and identity verification, and the demand for centralized platforms that simplify insurance management for both customers and administrators.

The problems identified throughout our study can be summarized as follows: a heavy dependence on traditional administrative procedures, limited real-time communication between insurance providers and their customers, and the absence of a unified mobile platform that integrates policy management, payment processing, claims submission, and biometric identity verification within a single application.

To tackle these problems, this project sets four main objectives. The first is the design and development of a cross-platform mobile insurance application for Android and iOS using Flutter. The second is the implementation of a secure backend architecture covering authentication, policy management, claims processing, and payment integration. The third is the integration of a biometric face verification service powered by artificial intelligence to strengthen identity validation. The fourth is the development of a web-based administrative dashboard for centralized monitoring and management of all insurance operations.

The work followed a structured methodological framework that begins with the study and analysis of existing insurance systems and digital solutions, followed by UML-based system modelling using use case, class, and sequence diagrams, then full-stack implementation using modern web and mobile technologies, and finally testing and validation of all developed functionalities.

This document is structured into four chapters. The first chapter introduces the fundamentals of insurance and mobile technologies. The second chapter reviews the state of the art and existing insurance solutions. The third chapter presents the analysis and design of the proposed system. The fourth chapter covers the implementation and technical architecture of the developed application.

The growing adoption of smartphones and digital financial services confirms that the demand for intelligent and user-friendly insurance platforms will only continue to increase. This project seeks to contribute to this evolution by delivering a modern mobile insurance system that brings together usability, security, scalability, and real-time service management within one cohesive platform.

1. Chapter 1: Basic Concepts and Types of Insurance

1.1 Introduction

Insurance is one of the most important financial tools designed to protect individuals and their property from unexpected financial risks. In light of technological advancements and digital transformation, the insurance sector must adopt modern methods to ensure faster and more secure services for policyholders.

In this chapter, we will review the general concepts of insurance, the different types of insurance with a focus on auto insurance the digital transformation in this sector, and the current solutions available

1.2 General Concepts of Insurance

1.2.1 Definition of Insurance

A contract between two parties, one of whom is called the insurer (the insurance company) and the other the client (the insured), in which the insurance company undertakes to pay the insured financial compensation in the event of an accident or if a risk specified in the contract materializes, in exchange for premiums paid by the insured to the insurance company [1].

1.2.2 Types of Insurance

Insurance aims to protect individuals and companies from financial risks by compensating them for losses resulting from unforeseen events. Insurance is divided into several types, including:

- **Life insurance:** Covers risks associated with death or disability and is designed to provide financial protection for beneficiaries.
- **Health insurance:** Covers the costs of treatment, hospitalization, and medication.
- Property Insurance: Protects homes, offices, and other property from damage or theft.
- **Liability Insurance**: Protects an individual or a company from having to pay compensation when they cause harm to others.
- **vehicle Insurance:** Covers damage to the vehicle, passenger injuries, liability to third parties, and sometimes theft and fire [1].

1.3 Vehicle insurance

1.3.1 Definition

Insurance is generally considered a method or system for managing risks. It is used by insurance companies to guarantee the rights of policyholders through a document known as an insurance contract. This document is essential for drivers, as it allows them to obtain coverage for their vehicles against various risks and ensures compensation in case of an accident. It also includes all the necessary information to identify both the driver and the vehicle, as well as details of the provided coverages [2].

Insurance can also be defined as a system that enables individuals or legal entities to protect themselves against the financial and economic consequences of specific risks such as fire, theft, or death. In the event that a risk covered by the insurance contract occurs, the insurer is required to provide financial compensation either to the insured, a third party, or a beneficiary, particularly in life insurance. In return, the insured pays a sum of money known as a premium, which is determined based on the level of risk and the operating costs of the insurance company [3].


The Oriental Insurance Company Limited
MOTOR INSURANCE CERTIFICATE CUM POLICY SCHEDULE
PRIVATE CAR PACKAGE POLICY - ZONE A

Figure 1.1 : Car insurance policy

1.3.2 General information about insurance

After defining insurance, it is important to address the key concepts associated with this field, as they help in understanding the various aspects that underpin the insurance system:

- **Accident:** An unexpected and unforeseen event that may result in property damage or bodily injury.
- **Insurer:** An insurance company that agrees to compensate for losses in accordance with the terms specified in the contract in exchange for the payment of premiums.
- **Insured:** is the person who enters into the insurance contract and benefits from the insurance coverage for the vehicle.
- **Beneficiary:** A person entitled to receive compensation, whether the insured or those affected by the accident.
- **Insurance Policy:** A legal agreement designed to protect the driver from damages and losses that may be caused to third parties while operating the vehicle.
- **Guarantee:** This represents the insurance company's commitment to bear the costs and losses resulting from a specific incident as stipulated in the contract.
- **Compensation:** The amount of money paid by the insurance company to cover the damages resulting from the accident.
- **Expert:** A professional appointed to assess the extent of the damage and estimate the amount of compensation following an accident.
- **Malfunction:** A mechanical or electrical fault that prevents the vehicle from operating or running normally.
- **Theft:** Means the unlawful taking of a vehicle or one of its accessories without the owner’s consent.
- **Insured Event:** An accident or incident that triggers the insurance coverage during the term of the policy.
- **Insured Vehicle:** The vehicle specified in the insurance policy and covered by the insurance policy.
- **Damage:** Any material, physical, or emotional loss that may be sustained by the insured or the vehicle.

- **Due Date:** The date specified for payment of the insurance premium or renewal of the policy.
- **Loss of Coverage:** The insured's loss of the right to compensation due to failure to comply with the terms of the policy or applicable laws [2], [3].

1.3.3 Types of insurance guarantees

Guarantees are among the most fundamental elements of an insurance contract, highlighting the various coverages that the company is obligated to provide to the insured to protect them from potential risks. These guarantees are divided into mandatory and optional, providing additional protection for the vehicle and driver, and include:

- **Civil Liability Guarantee:** This is considered mandatory coverage, under which the insurance company covers property damage and bodily injury that the driver may cause to others while driving, as well as medical expenses and losses resulting from traffic accidents [4].
- **Comprehensive Coverage:** This coverage provides comprehensive protection for the insured vehicle, covering damages resulting from collisions, rollovers, or various other accidents, regardless of whether the insured is at fault [5].
- **Fire and Theft Coverage:** This coverage is designed to cover losses resulting from the vehicle being damaged by fire or stolen, either in whole or in part, and also includes certain accessories and genuine replacement parts for the vehicle [5].
- **Glass Breakage Coverage:** This coverage includes the repair or replacement of damaged vehicle glass, such as the windshield, rear window, side windows, and mirrors [4].
- **Natural Disaster Coverage:** Provides compensation for damage resulting from natural disasters such as floods, storms, and earthquakes, which may cause significant losses to the insured vehicle [6].
- **Roadside Assistance Coverage:** This type of coverage ensures that assistance services are provided to the driver in the event of a vehicle breakdown, such as towing the vehicle or repairing minor breakdowns on-site [6].
- **Personal Accident Coverage:** This coverage provides compensation to the driver and passengers in the event of injury or death resulting from a traffic accident, thereby enhancing the level of protection for the people inside the vehicle [5].

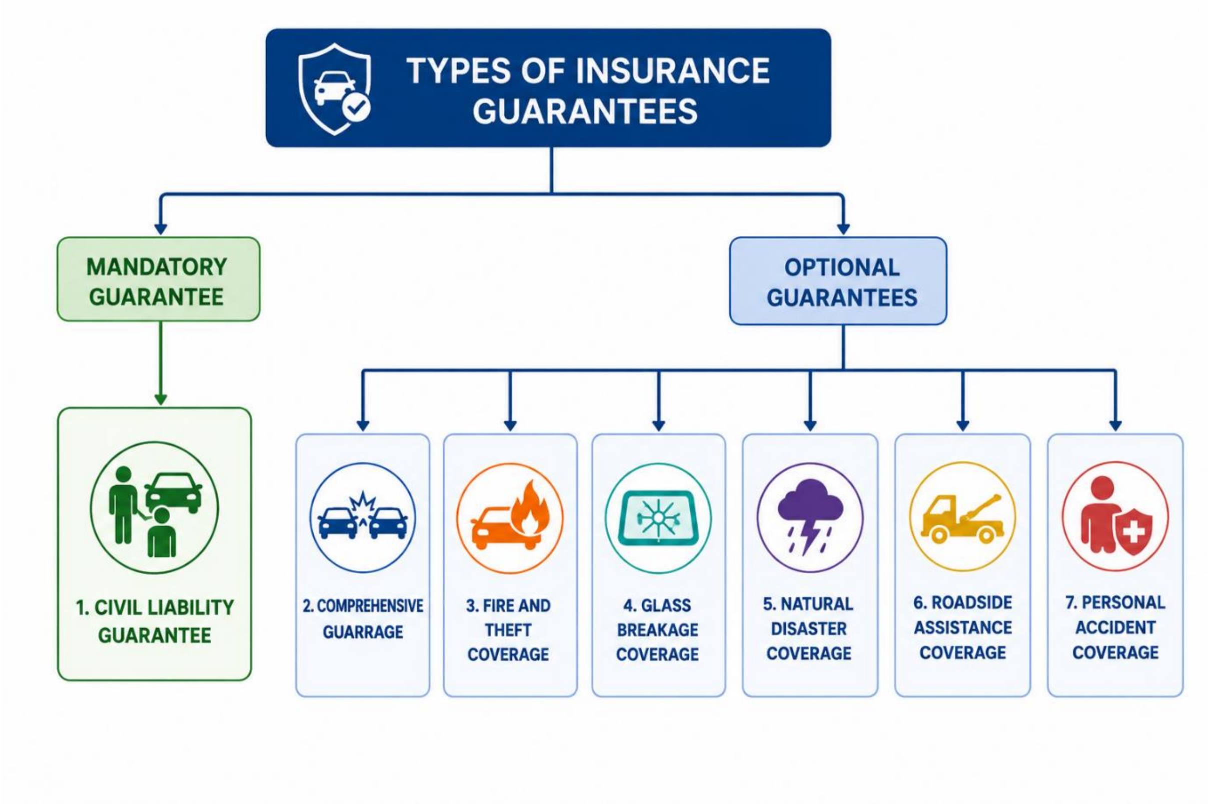


Figure 1.2 : Types of insurance guarantees

1.3.4 An Overview of the Insurance Market in Algeria

The Algerian insurance market is dominated by several public companies that play a key role in providing insurance services, especially in the automotive sector.

1.3.4.1 SAA - National Insurance Company

It is the largest insurance company in Algeria in terms of market share. It was established in 1963 and is government-owned. It offers all types of insurance, especially vehicle insurance, and has a wide network of branches throughout the country [7].

1.3.4.2 CAAT - Algeria Insurance and Reinsurance Company

A public company specializing in damage insurance, particularly vehicle and transport insurance. It also boasts extensive experience dating back to Algeria's independence and possesses a strong network of agencies within the country [7].

1.3.4.3 CAAR – Algerian Insurance and Reinsurance Company

A national company specializing in property insurance and major risk coverage, in addition to vehicle insurance, plays a role in reinsurance within Algeria [7].

1.4 Mobile Applications

1.4.1 Definition of Mobile Applications

Mobile applications are software programs designed to run on mobile devices such as smartphones and tablets. These applications provide users with a variety of services and functions via mobile devices connected to wireless networks. Mobile applications are developed for mobile operating systems such as Android and iOS, and are widely used in many fields, including education, healthcare, banking, e-commerce, and insurance [8].

Mobile applications are specifically designed to improve accessibility, enhance user experience, and provide services anytime and anywhere using mobile technologies [9].

Figure 1.3 : Smartphone showing apps

1.4.2 Characteristics of Mobile Applications

Mobile applications possess characteristics that distinguish them from traditional desktop applications. They are portable, user-friendly, and optimized for touch interfaces. Depending on their function, they can operate online or offline, and often provide immediate interaction with users [8].

Furthermore, mobile applications can access device permissions such as GPS, camera, sensors, and notifications, enhance the user experience and enable access to advanced features.

1.4.3 Types of Mobile Applications

Mobile applications can be classified into several types according to how they are developed and operated, and some of the most important of these types are:

- **Native Applications**: Applications developed specifically for a single operating system such as Android or iOS using platform-specific languages. They provide high performance and full access to device functionalities [9].
- **Web Applications:** Mobile-friendly websites accessed through web browsers without installation. They require internet connectivity and provide limited access to device features [9].
- **Hybrid Applications:** are software solutions that merge characteristics of both native and web applications. They are built using web technologies and then packaged within native containers, which enables them to operate across different platforms [9].

1.4.4 Mobile Operating Systems

Mobile operating systems are software platforms that manage and control the hardware and software resources of mobile devices. The most widely used mobile operating systems include Android and iOS [10], [11]:

- **Android:** Developed by Google, Android is an open-source operating system widely used on smartphones from different manufacturers. It provides flexibility, customization, and a large ecosystem of applications [10].

Figure 1.4 : Android logo

- **IOS**: is a mobile operating system created by Apple for its own devices like iPhones and iPads. It is designed to provide a secure and stable environment, while ensuring smooth and optimized performance for users [11].

Figure 1.5 : IOS Logo

1.4.5 Role of Mobile Applications in Modern Systems

Mobile applications are now considered essential elements of modern digital systems, and are widely adopted in various sectors such as healthcare, banking, education, transportation, e-commerce, and insurance [12].

They enhance communication, facilitate access to services, reduce operational costs, and improve efficiency. In organizational contexts, mobile applications enable companies to deliver real-time services and increase customer satisfaction through more personalized and accessible solutions [8].

1.4.6 Mobile Applications in Insurance Systems

Mobile applications have revolutionized the insurance industry by offering digital solutions that enhance efficiency, facilitate access, and enrich the customer experience. Previously, traditional insurance procedures relied heavily on in-person visits and paperwork, resulting in lengthy processing times. Today, with the adoption of smartphone technologies, insurance services are more dynamic and more focused on meeting user needs [12].

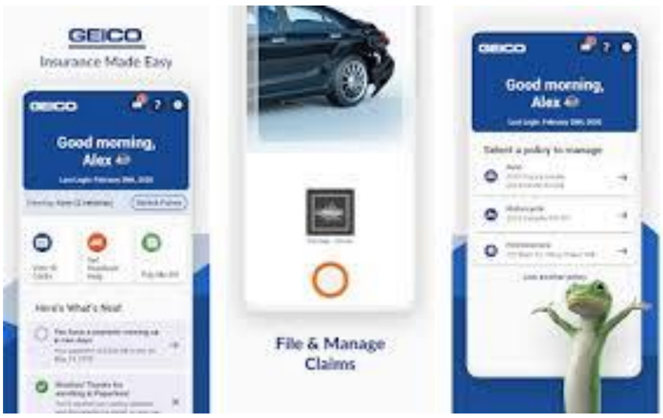


Figure 1.6 : An Example of Insurance mobile app

1.5 Conclusion

This chapter has provided the necessary theoretical foundations for understanding the main concepts related to insurance systems and mobile technologies. It introduced the basic principles of insurance, with a particular focus on vehicle insurance, and highlighted its key types, guarantees, and role within the insurance market. In addition, it explored the concept of mobile applications and their importance in modern digital systems, as well as their different types, characteristics, and operating environments.

These concepts form an essential basis for understanding how digital transformation is reshaping the insurance sector and improving service delivery. Consequently, this theoretical background serves as a solid foundation for the design and development of the proposed mobile insurance application, which will be further detailed in the following chapters.

2. Chapter 2: State of the Art and Existing Insurance Solutions

2.1 Introduction

This chapter builds upon the concepts presented in Chapter 1 and focuses on current mobile and online insurance solutions, particularly in the automotive insurance sector. It analyzes existing applications and platforms to understand how insurance services are presented digitally today.

This study aims to evaluate the key features offered by these applications, identifying their strengths and weaknesses. This analysis helps to highlight gaps in current solutions and determine the need for an improved mobile insurance application that is more efficient, user-friendly, and better suited to user needs.

2.2 Analysis of Existing Insurance Solutions

Research indicates a fundamental shift in the insurance sector, with increasing reliance on digital and mobile technologies to facilitate service delivery and customer communication. Studies on mobile e-commerce in the insurance industry show that digital platforms enable users to complete various tasks, such as managing insurance policies, accessing information, and submitting service requests, all through their mobile devices [13].

However, these solutions depend on the degree to which the technology meets users' needs and requirements. Studies indicate that poorly designed insurance applications can negatively impact their effectiveness and usability, leading to decreased user satisfaction and inefficient service delivery [13].

Therefore, while existing insurance systems provide essential digital services, there is still a need for more optimized and user centred mobile applications that better support real-time interaction, simplicity, and complete insurance lifecycle management.

2.3 Existing Systems:

2.3.1 SAA Mobile Services (Algeria)

The Société Nationale assurance (SAA) provides insurance services in Algeria, including vehicle insurance. It has introduced digital services through its online platform, allowing users to consult insurance information and manage certain services remotely [14].

However, the system is still partially digitized, as many procedures such as claims submission or document validation often require physical visits to agencies. This limits the full usability of its digital services and affects user convenience [14].

Figure 2.1 : SAA (Société Nationale assurance) – Algeria

Based on our review of the platform's publicly available services, the following points were identified:

2.3.1.1 Advantages:

- Large national insurance company with strong market presence in Algeria.
- Wide network of physical agencies across the country.
- Provides basic online services such as policy information and insurance quotations.
- Allows users to access general insurance information remotely.
- Facilitates initial contact without the need to visit agencies.

2.3.1.2 Disadvantages:

- Services are only partially digitalized.
- Lack of a fully functional mobile application for complete insurance management.
- Claims submission and document validation still require physical visits.
- Limited real-time tracking of insurance requests.
- User experience is not fully optimized for modern mobile interfaces.

- No full end-to-end digital insurance process (from subscription to claim settlement).

2.3.1.3 Technologies And Services Used:

- Web-Based Insurance Platform
- Mobile Digital Services
- Online Customer Management
- Insurance Information System
- Authentication and Security Services
- Database Management Systems

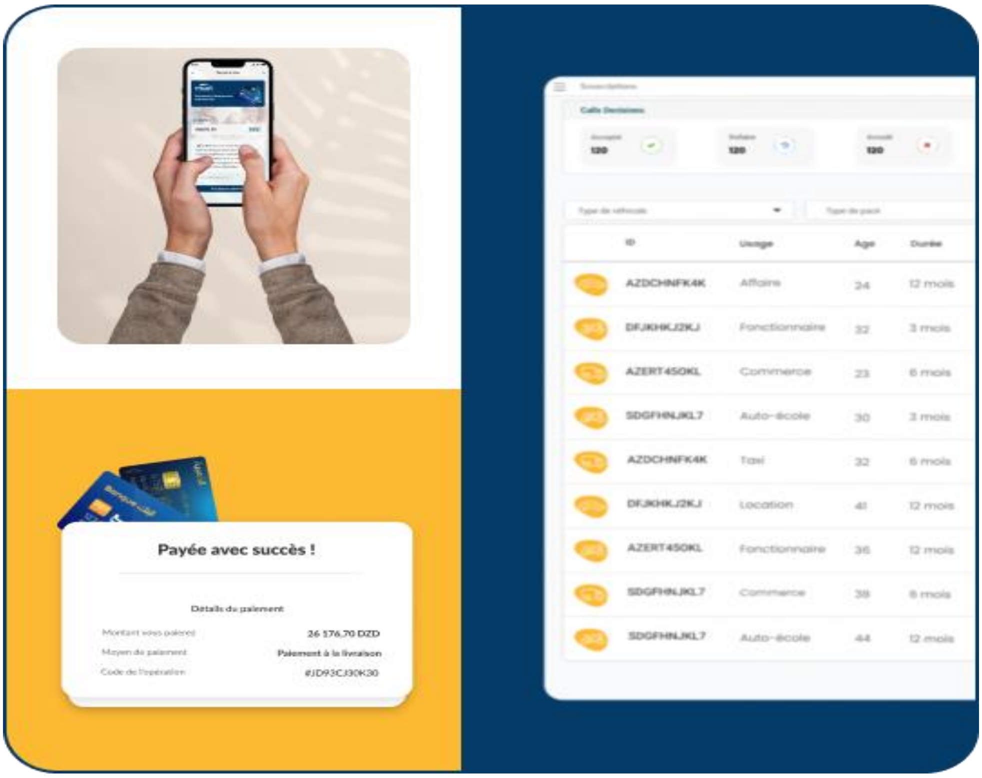


Figure 2.2 : SAA Mobile Services Interface

2.3.2 CAAT (Algerian Insurance and Transport Company)

The Algerian General Insurance Company (CAAT) specializes in transport and vehicle insurance. The company aims to provide insurance solutions for individuals and businesses, focusing on motor vehicle and industrial risks [7].

The company has an online presence that allows users to access basic information and insurance services. However, most processes are still partially manual and require physical interaction with agencies.

Figure 2.3 : CAAT (Algerian Insurance and Transport Company)

Based on our review of the platform's publicly available services, the following points were identified:

2.3.2.1 Advantages:

- Strong expertise in vehicle and transport insurance.
- Long experience in the Algerian insurance market.
- Provides basic online information and services.
- Wide agency network in Algeria.

2.3.2.2 Disadvantages:

- Limited digital transformation compared to international standards.
- Lack of advanced mobile application for full insurance management.
- Manual processing still required for claims and documents.
- No real-time tracking of insurance requests.

2.3.2.3 Technologies And Services Used:

- Web Information System
- Online Insurance Consultation
- Customer Data Management
- Digital Communication Services
- Administrative Processing Systems

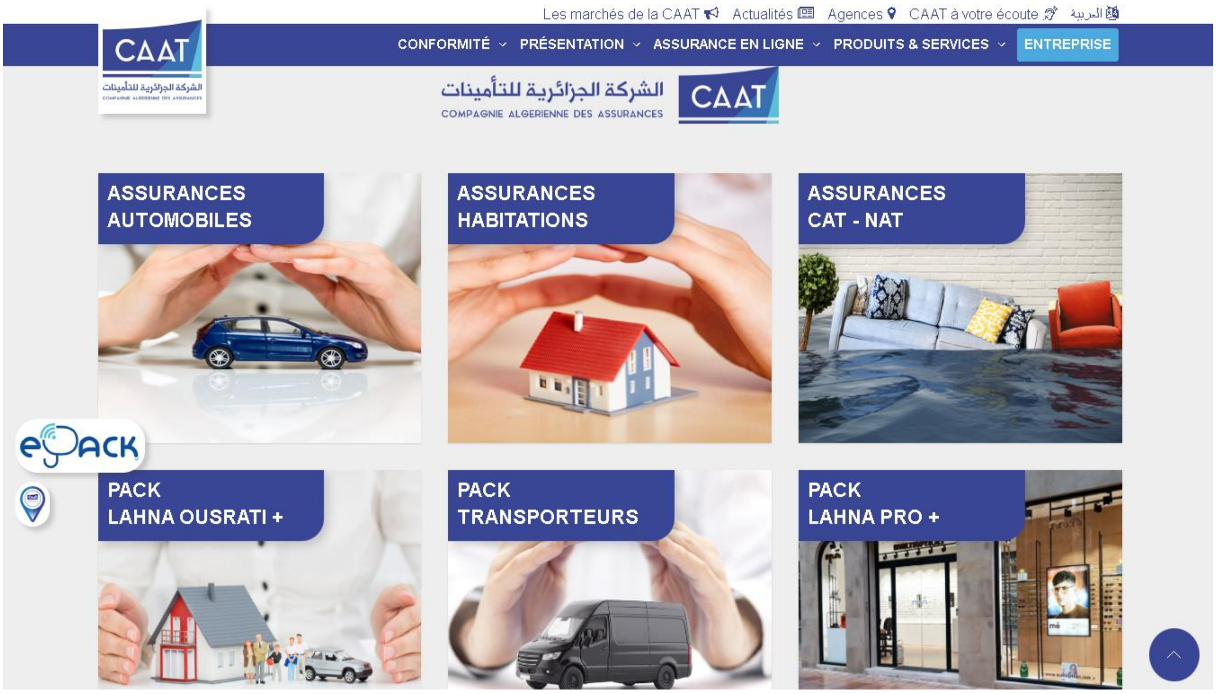


Figure 2.4 : CAAT Platform

2.3.3 CAAR (Algerian Insurance and Reinsurance Company)

CAAR is a national insurance and reinsurance company in Algeria. It mainly focuses on property insurance and large-scale risks, in addition to vehicle insurance services [7].

Although CAAR plays an important role in the Algerian insurance market, its digital services remain limited and traditional procedures are still widely used.

Figure 2.5 : CAAR (Algerian Insurance and Reinsurance Company)

Based on our review of the platform's publicly available services, the following points were identified:

2.3.3.1 Advantages:

- Specialization in major risks and insurance coverage.
- Strong role in reinsurance sector in Algeria.
- Offers various insurance products including automotive coverage.
- Established and reliable institution.

2.3.3.2 Disadvantages:

- Limited digital and mobile services
- Heavy reliance on traditional paper-based procedures
- Lack of user-friendly mobile applications
- Slow adoption of full digital transformation

2.3.3.3 Technologies And Services Used:

- Insurance Management System
- Database Systems
- Web Service Interfaces
- Administrative Information System
- Customer Support Services

Figure 2.6 : CAAR Platform

2.4 Comparative Analysis:

Criteria	SAA	CAAT	CAAR
Mobile Application Availability	Partial	Limited	Limited
Online Services	Basic Services	Basic Services	Basic Services
Claim Submission	Partially Digital	Mostly Manual	Mostly Manual
Real-Time Tracking	Limited	Not Available	Not Available
User Interface	Moderate	Basic	Basic
Document Upload	Limited	Limited	Limited
Notifications System	Limited	Not Available	Not Available
Agency Dependence	High	High	High

Digital Transformation Level	Medium	Low	Low
Customer Experience	Moderate	Moderate	Moderate

Table 2.1 : Comparison Between Existing Insurance Applications

2.5 Analysis of the Comparison

The comparison shows that there are clear differences in the level of digital development among Algerian insurance platforms. While some companies have started introducing basic online services, the overall digital transformation remains limited and not fully integrated.

In particular, most existing systems still rely on traditional procedures and physical agency interactions for key operations such as claim submission, document verification, and final processing. Although some digital tools have been introduced, their functionality is still restricted and does not provide a complete mobile insurance experience.

This comparison highlights the need for a modern and fully integrated mobile insurance application capable of improving user experience, simplifying insurance procedures, and enabling faster and more efficient service delivery.

2.6 Conclusion

This chapter examined existing vehicle insurance platforms by analysing their primary features, strengths, and weaknesses. While some advancements have occurred in digital transformation, most current systems remain only partially digitized and depend significantly on traditional procedures and in-person agency interactions, which reduces efficiency and hampers the user experience. This highlights the pressing need for a modern, fully integrated mobile insurance application that enhances service quality, streamlines processes, and delivers faster, more user-centric services. The insights from this chapter will inform the next chapter, which presents the design of the proposed system through UML diagrams.

3. Chapter 3: Analysis and Design of the Proposed System

3.1 Introduction

This chapter covers the design phase of the proposed mobile insurance application. Building on the analysis conducted in the previous chapters, this stage aims to translate the identified requirements into a clear and structured system design.

System design is a critical step in the software development process, as it bridges the gap between the theoretical analysis and the practical implementation. A well-thought-out design not only reduces development errors but also ensures that the final product meets the expected functional and non-functional requirements.

The aim is to represent the different functionalities of the application through UML diagrams. Three main types of diagrams are used: use case diagrams to show how actors interact with the system, class diagrams to define the static structure and relationships between the different components, and sequence diagrams to illustrate the dynamic behavior of the system when handling specific processes.

These diagrams provide a comprehensive and readable blueprint of how the application is expected to operate prior to the implementation phase. They also serve as a reference point throughout the development process, ensuring consistency between the initial design decisions and the final built system.

3.2 Modeling Methodology: UML

3.2.1 Definition of UML

The Unified Modeling Language (UML) is a standardized visual modeling language widely used in software engineering to specify, design, and document software systems. It provides a set of graphical notations that allow developers and analysts to represent the structure and behavior of a system in a clear and unambiguous way [15].

UML was adopted in this project as the primary modeling tool due to its widespread use in object-oriented software development and its ability to capture both the static and dynamic aspects of a system. It serves as a common language between all stakeholders involved in the development process, ensuring that the system is well understood before any implementation begins [16].

3.2.2 Importance of UML

UML plays a crucial role in software development because it:

- Improves system understanding and visualization
- Facilitates communication between developers and clients
- Helps detect design errors early in the development process
- Provides a standardized modeling approach
- Supports documentation of complex systems [16].

3.2.3 Types of UML Diagrams Used

In this project, several UML diagrams are used to model the mobile insurance system:

- **Use Case Diagrams**, which define the functional boundaries of the system and describe the interactions between the different actors and the available features of the application [17].
- **Class Diagrams**, which represent the static structure of the system by identifying the main entities, their attributes, and the relationships that exist between them [17].
- **Sequence Diagrams**, which illustrate the dynamic behavior of the system by showing how objects interact with each other over time in response to specific user actions [16].

Together, these diagrams provide a comprehensive and structured view of the proposed mobile insurance application, forming a solid foundation for the implementation phase that follows.

3.3 Identification of Actors

In UML modeling, an actor represents any entity that interacts with the system from the outside, whether it is a human user, an organization, or an external system. Identifying actors is a fundamental step in the design process, as it helps define the system boundaries and clarify the roles of each participant involved in the application [15].

Based on the requirements analysis conducted in the previous chapters, eight actors have been identified for the proposed mobile insurance application, classified into primary human actors, and secondary system actors.

3.3.1 Administrator

The administrator is responsible for managing and supervising the entire system. This actor has full control over the application and ensures its proper functioning at all times.

Main responsibilities:

- Manage users and accounts
- Validate insurance subscriptions
- Manage insurance contracts
- Monitor claims and notifications
- Supervise system activities

3.3.2 Visitor (Guest User)

The visitor is a non-authenticated user who can access the application to browse available insurance services and consult general information prior to registration.

Main functionalities:

- View insurance offers
- Consult application services
- Create a new account

3.3.3 Client / Insured User

The insured user is an authenticated client recognized by the system. This actor can subscribe to insurance services, manage existing policies, and declare claims directly through the mobile application.

Main functionalities:

- Login to the application
- Subscribe to insurance policies
- Submit insurance claims
- Upload required documents
- Track insurance requests
- Renew insurance contracts

3.3.4 Insurance Agent

The insurance agent is responsible for handling insurance-related operations and providing assistance to clients throughout the subscription process.

Main responsibilities:

- Verify client information
- Manage insurance contracts
- Process insurance requests
- Communicate with insured clients

3.3.5 Claim Service Agent

The claim service agent oversees all accident and claim-related operations. This actor reviews submitted claims and coordinates the necessary compensation procedures.

Main responsibilities:

- Review claim requests
- Verify uploaded documents
- Manage claim files
- Process compensation procedures

3.3.6 Expert

The expert is responsible for conducting on-site inspections of damaged vehicles and evaluating accident-related damages in order to determine the appropriate compensation amount.

Main responsibilities:

- Examine damaged vehicles
- Evaluate accident damages
- Prepare expert reports
- Estimate compensation costs

3.3.7 Payment System

The payment system is a secondary actor that handles all financial transactions within the application, including insurance premium payments and compensation-related operations.

3.3.8 Printer

The printer is a secondary actor that enables users to generate printed copies of insurance contracts and other relevant documents when required.

The following table summarizes all identified actors and their roles within the system:

Actor	Type	Main Role
Administrator	Primary	Manages and supervises the entire system
Visitor	Primary	Browses services and creates an account
Client / Insured User	Primary	Subscribes, submits claims, manages policy
Insurance Agent	Primary	Processes subscriptions and assists clients
Claim Service Agent	Primary	Reviews and processes claim requests
Expert	Primary	Evaluates damages and prepares reports
Payment System	Secondary	Handles financial transactions
Printer	Secondary	Generates printed documents

Table 3.1 : identified actors and their roles

3.4 Identification of Messages

After identifying the different actors of the proposed mobile insurance system, it is necessary to describe the various messages exchanged between the actors and the system. These messages represent the interactions and requests performed during the use of the application functionalities.

3.4.1 Messages Sent by the System

The system generates and sends various responses and notifications to users depending on the actions performed. The main messages sent by the system include:

- Display the list of users, insurance contracts, and claims
- Display registration and subscription forms
- Send notifications and alerts
- Confirm authentication and successful operations
- Display claim status and insurance information
- Generate insurance contracts and reports

3.4.2 Messages Received by the System

The system receives different types of requests from users and external actors throughout the lifecycle of the application. These include:

- Authentication requests (Login / Registration)
- Insurance subscription requests
- Claim submission requests
- Requests for consultation and information access
- Document upload requests
- Requests for adding, modifying, deleting, or viewing data
- Payment and contract renewal requests

The identification of these messages helps establish a clear picture of the communication flow between the system and its actors, and serves as a direct foundation for modeling the use case and sequence diagrams presented in the following sections.

3.5 Use Case Diagrams

Use case diagrams are among the most fundamental diagrams in UML modeling. They provide a high-level view of the system's functionalities by illustrating the interactions between the identified actors and the various use cases available within the application [15].

In the proposed mobile insurance application, the use case diagrams describe the main operations such as user registration, authentication, insurance subscription, claim submission, policy management.

The following sections present the global use case diagram of the proposed mobile insurance application, along with detailed descriptions of the principal use cases identified for each functional module. These descriptions provide a clear understanding of the interactions between the system and its actors.

3.5.1 Global Use Case Diagram

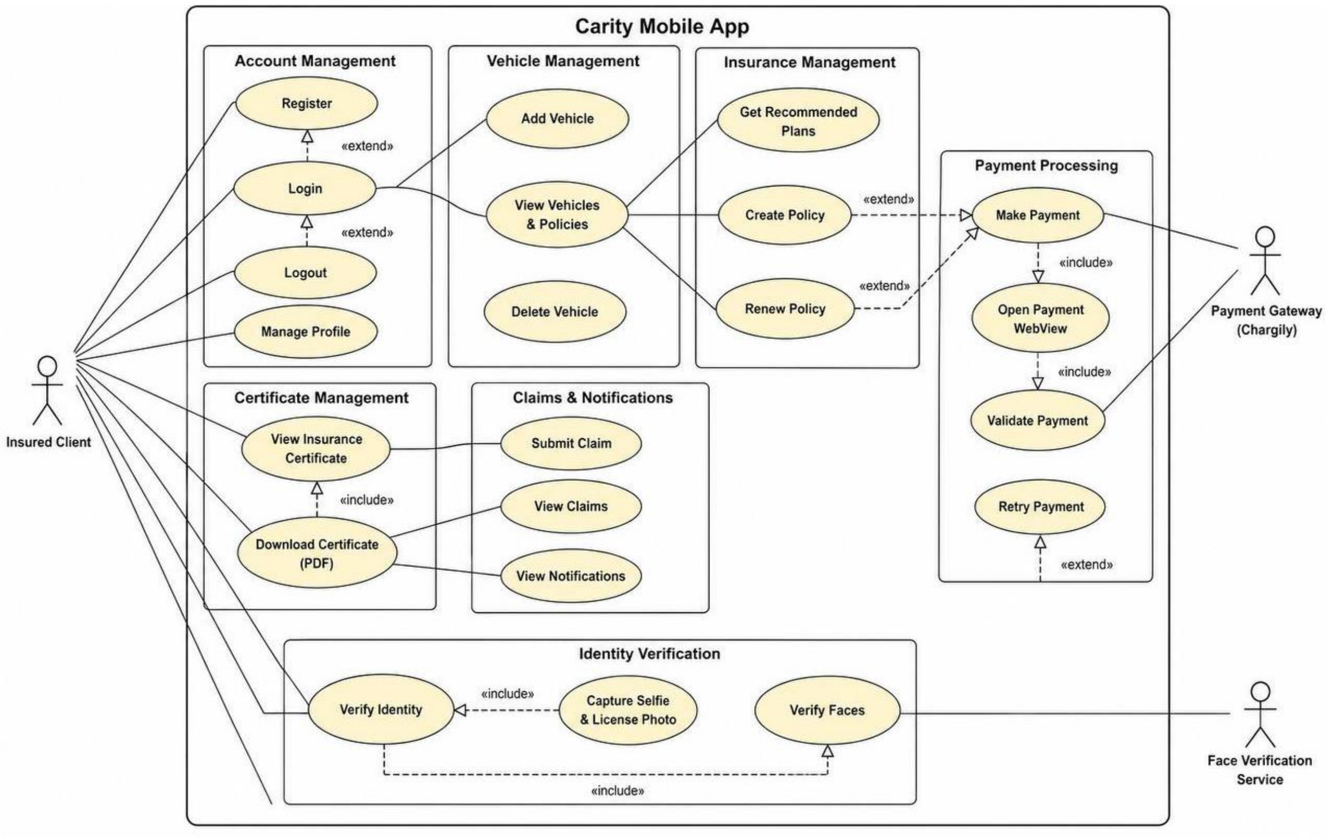


Figure 3.1 : Global Use Case Diagram (part 1)

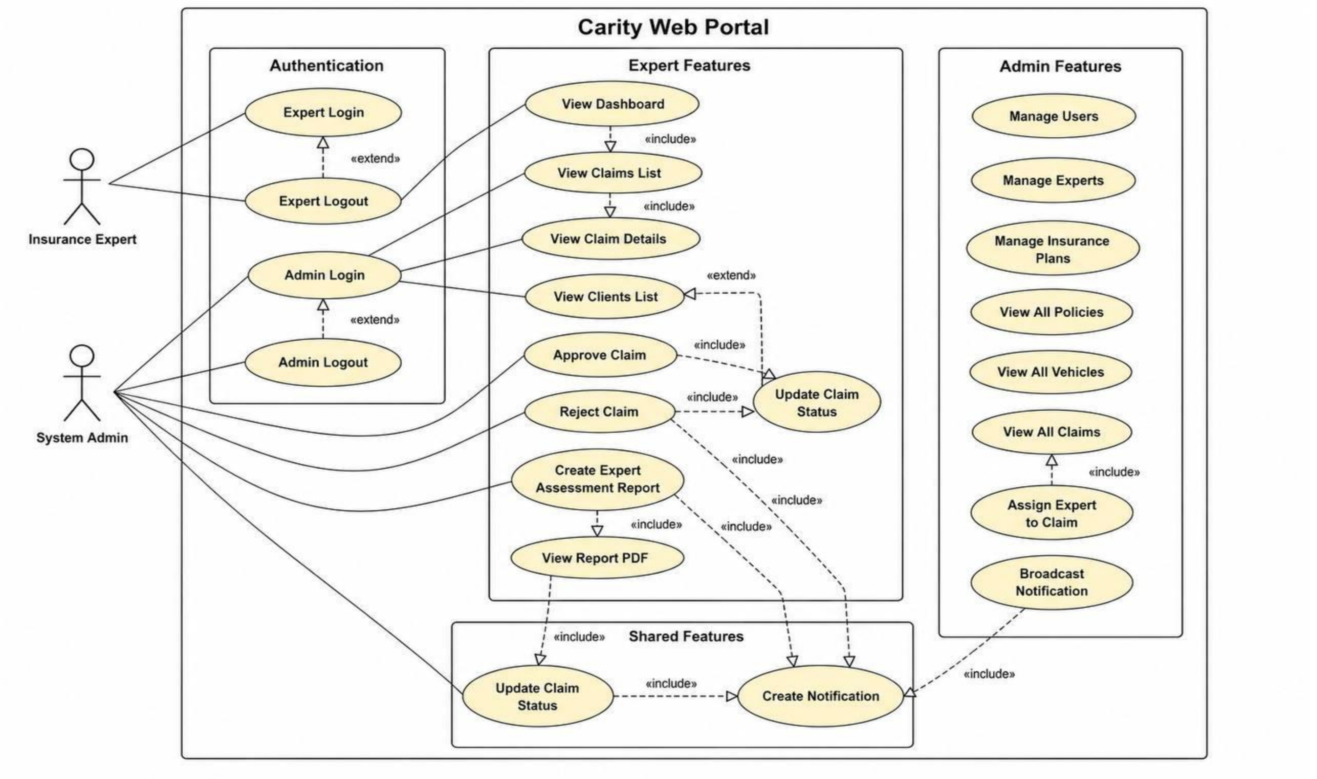


Figure 3.2 : Global Use Case Diagram (part 2)

Overview:

The diagram above illustrates the main interactions between the actors and the proposed mobile insurance application. It presents the different functionalities provided by the system, including authentication, insurance subscription, claim management, payment processing, and identity verification. It also highlights the communication between the system and external services, which are used to ensure security, reliability, and automation of insurance operations.

3.5.2 Detailed Use Case

In this section, after presenting the global use case diagram, a detailed description sheet is provided of the most important key use cases identified within the system:

- **User Registration Use Case**

Field	**Description**
Use Case	User Registration
Primary Actor	Visitor
Objective	Allow a new user to create an account in the system
Precondition	User does not already have an account
Postcondition	A new user account is created successfully
Main Scenario	1- User selects “Register” 2- System displays registration form 3- User enters personal information 4- System validates data 5- Account is created successfully
Alternative Scenario	4-1 System displays error message if data is invalid

Table 3.2 : User Registration Use Case Description Sheet

- **Login Use Case**

Field	Description
Use Case	Authentication / Login
Primary Actor	User
Objective	Allow user to access the system securely
Precondition	User must already be registered
Postcondition	User is authenticated and granted access
Main Scenario	1- User opens login page 2- User enters email & password 3- System sends credentials to backend 4- System verifies credentials 5- User is redirected to home page
Alternative Scenario	4-1 System shows error message for incorrect credentials

Table 3.3 : Login Use Case Description Sheet

- **Authentication Use Case**

Field	Description
Use Case	Authenticate Credentials (Internal)
Primary Actor	System (Backend Service)
Objective	Verify user credentials securely
Precondition	Login request received

Field	Description
Postcondition	User is authenticated or rejected
Main Scenario	1- System receives credentials 2- System checks database 3- System validates password hash 4- System returns authentication result
Alternative Scenario	3-1 Credentials are invalid → authentication failed

Table 3.4 : Authentication Use Case Description Sheet

- **Insurance Subscription Use Case**

Field	Description
Use Case	Insurance Subscription
Primary Actor	Client
Objective	Allow the user to subscribe to an insurance policy through the mobile application
Precondition	User must be authenticated
Postcondition	Insurance contract is created and stored in the system

Field	**Description**
Main Scenario	1- User selects “Subscribe Insurance” 2- System displays available insurance offers 3- User selects a policy type 4- User enters vehicle and personal information 5- System validates data 6- System confirms subscription and generates contract
Alternative Scenario	5-1 System rejects subscription if data is incomplete or invalid

Table 3.5 : Insurance Subscription Use Case Description Sheet

- **Claim Submission Use Case**

Field	**Description**
Use Case	Claim Submission
Primary Actor	Client
Objective	Allow the insured user to declare an accident and submit a claim
Precondition	User must be authenticated and have an active insurance policy
Postcondition	Claim is registered in the system and sent for processing

Field	Description
Main Scenario	1. User selects "Submit Claim" 2. System displays claim form 3. User enters accident details 4. User uploads required documents (photos, reports) 5. System validates submission 6. System stores claim and assigns status "Pending"
Alternative Scenario	5-1 System rejects submission if required documents are missing

Table 3.6 : Claim Submission Use Case Description Sheet

- **Policy Management Use Case**

Field	Description
Use Case	Policy Management
Primary Actor	Client / Administrator
Objective	Allow management of insurance policies (view, renew, update)
Precondition	User must be authenticated
Postcondition	Policy information is updated or displayed successfully
Main Scenario	1- User accesses policy section 2- System displays active policies 3- User selects a policy 4- User views details or requests renewal/update 5- System processes request and updates data
Alternative Scenario	4-1 System displays error if policy is expired or invalid

Table 3.7 : Policy Management Use Case Description Sheet

3.6 Class Diagram

3.6.1 Identification of Classes and Attributes

The identification of classes and attributes represents an important phase in the design of the proposed mobile insurance application. It consists of determining the principal entities involved in the system and defining the data associated with each entity.

Based on the system requirements and functionalities, several classes have been identified, including user management, insurance policies, claims management, payments, notifications, and identity verification services.

Each class contains attributes and methods that define its characteristics and responsibilities within the system. For example, **the User class** contains personal information such as name, email, phone number, and address, while **the Policy class** stores insurance-related information such as policy type, premium, and validity period.

The identification of these classes helps organize the system structure and serves as the foundation for the global class diagram presented in the following section.

3.6.2 Relationships Between Classes

Relationships between classes define how the different components of the system interact and communicate with each other. These relationships help describe the logical organization of the application and illustrate data exchange between system entities.

Several types of relationships are used in the proposed system, including:

- **Association:** Represents communication or interaction between classes.
- **Inheritance:** Defines hierarchical relationships between parent and child classes.
- **Composition:** Indicates strong dependency between classes where one object cannot exist without another.
- **Dependency:** Represents temporary usage relationships between system services.

For example, a User can own multiple insurance policies, while each Policy is associated with one specific vehicle. Similarly, a Claim may generate an Insurance Certificate and contain one or more Expert Reports.

These relationships contribute to creating a coherent and organized software architecture.

3.6.3 Global Class Diagram

The class diagram represents the static structure of the proposed mobile insurance application, showing its main classes, attributes, methods, and relationships. It covers the core components of the system such as authentication, policy management, claim processing, payments, notifications, and identity verification, while also illustrating interactions with external services supporting system functionality.

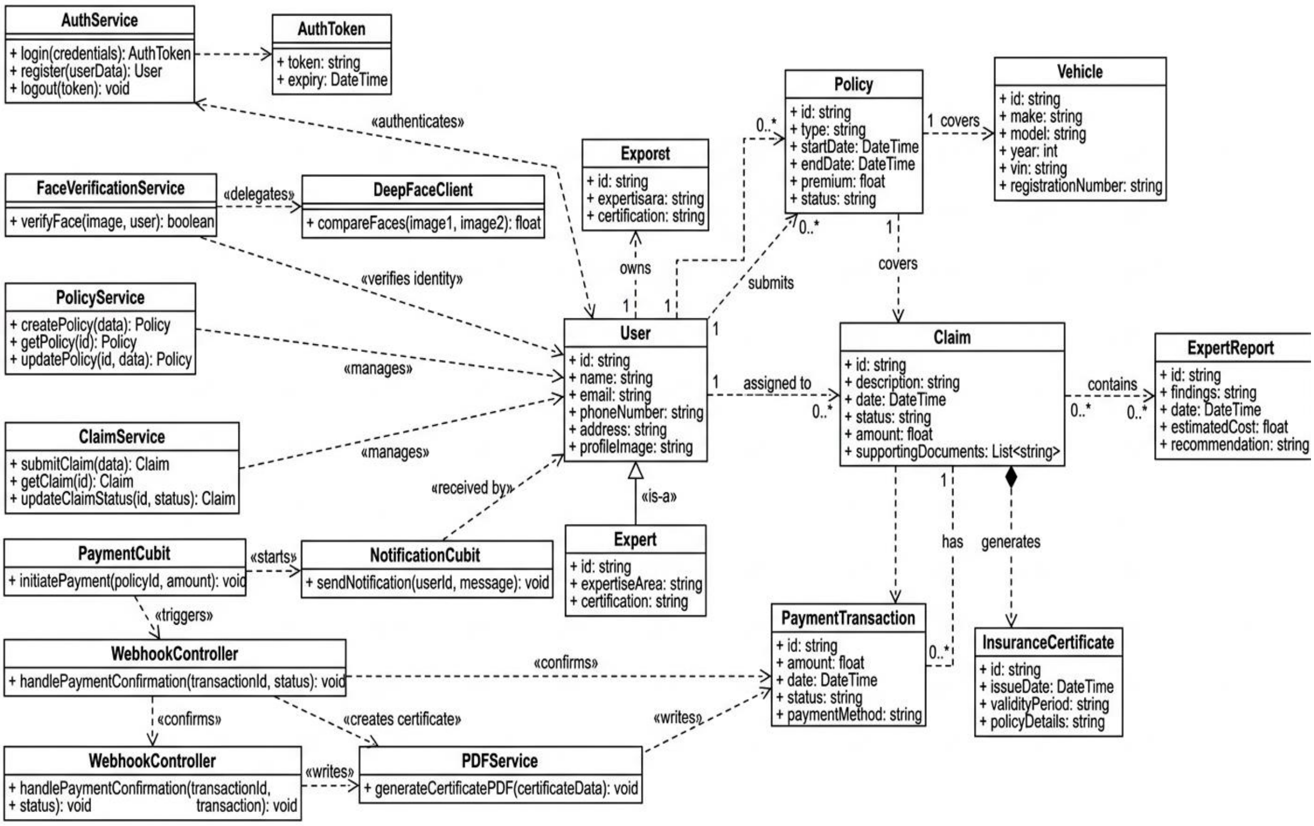


Figure 3.3 : Global Class Diagram of the Proposed Mobile Insurance System

3.6.4 Class Diagram Description

The proposed mobile insurance system's categorization structure is built around several core entities representing the application's main functions, including user management, insurance policies, claims processing, payments, notifications, and identity verification services. This diagram provides a comprehensive view of the system architecture.

At the center of the system is **the User class**, which acts as the main parent entity for different types of users such as Client, Administrator, Insurance Agent, Claim Service Agent, and Expert. These subclasses inherit shared attributes from the User class, including personal information, authentication data, and contact details, while also defining specific roles and responsibilities according to each user type.

The Client class is strongly associated with the Insurance Policy class through a one-to-many relationship, meaning that a single client can subscribe to multiple insurance policies at the same time. Each insurance policy is linked to a specific Vehicle class, which stores detailed information about the insured vehicle such as registration number, model, and other relevant specifications.

The Claim class is directly connected to both the Client and Insurance Policy classes, representing the process of accident declaration and claim management within the system. Each claim may generate one or more Expert Reports, which are used to evaluate damages, verify accident details, and determine the appropriate compensation amount.

The Payment class is associated with the Insurance Policy class, where each policy may include multiple payment transactions related to subscriptions, renewals, or compensations. This ensures proper financial tracking and management of all insurance operations within the system.

In addition, **the Notification class** is linked to the User class in a one-to-many relationship, allowing the system to send alerts and updates regarding policy status, claim processing, payment confirmation, and general system activities. This improves communication between the system and users.

Finally, external services such as Authentication Service and Identity Verification (DeepFace Service) are integrated into the system architecture. These services interact with the User and Claim processes to ensure secure access, prevent fraud, and improve the reliability of identity validation mechanisms.

3.7 Sequence Diagram

3.7.1 Definition

A Sequence Diagram is a UML behavioral diagram that describes how objects within a system interact in a time-ordered sequence. It focuses on showing the flow of messages exchanged between different components in order to achieve a specific functionality or system process.

This type of diagram emphasizes the chronological order of interactions, starting from an initial request and continuing step by step until the final result is achieved. Each participating object is represented by a lifeline, while the communication between objects is illustrated using messages exchanged over time.

Sequence Diagrams are widely used during the system design phase because they help visualize how processes are executed, clarify the order of operations, and show how different components collaborate to complete a task. They also assist in refining use cases and improving the overall understanding of system behavior and interaction flow.

3.7.2 Authentication Sequence Diagram

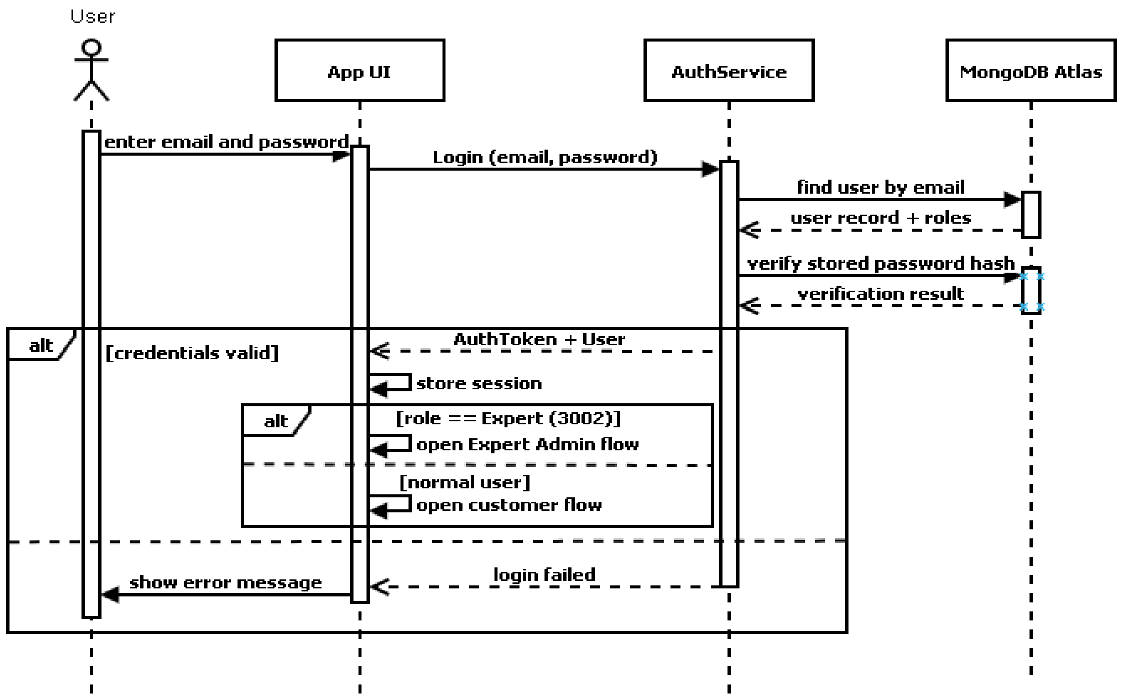


Figure 3.4 : Authentication Sequence Diagram

Explanation:

The previous sequence diagram models the interaction triggered when a user attempts to log into the application. The participants involved in this process are the User, the Application Interface, the Authentication Service, and the MongoDB Atlas database.

The process begins when the User enters their email and password through the application interface. The Interface then sends a login request to the Authentication Service, which communicates with the database to retrieve the corresponding user record and associated roles.

After retrieving the user information, the Authentication Service verifies the stored password hash and determines whether the provided credentials are valid. An Alt combined fragment is used to represent the possible outcomes of the authentication process:

• **[Credentials valid]:** The Authentication Service generates an authentication token and creates a secure user session. The system then redirects the user according to their assigned role. If the role corresponds to an Expert Admin, the Expert Admin flow is opened; otherwise, the normal customer flow is displayed.

• **[Credentials invalid]:** The Authentication Service returns a login failed response, which is propagated back to the User through the Interface in the form of an error message.

3.7.3 Policy Creation/Payment Process Sequence Diagram

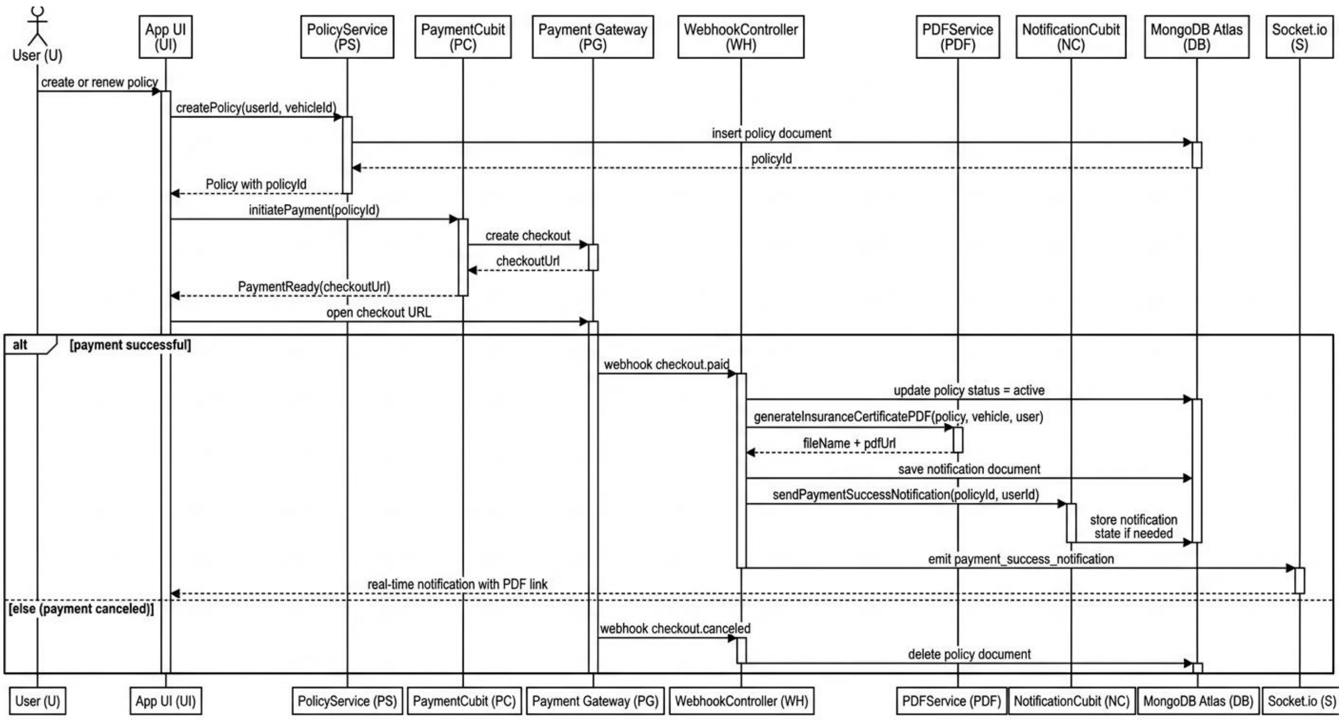


Figure 3.5 : Policy Creation/Payment Process Sequence Diagram

Explanation:

The following sequence diagram models the interaction triggered when a user creates or renews an insurance policy through the mobile application. The participants involved in this process are the User, the Application Interface, the Policy Service, the PaymentCubit, the Payment Gateway, the Webhook Controller, the PDF Service, the NotificationCubit, the MongoDB Atlas database, and the Socket.io service.

The process begins when the User requests the creation or renewal of a policy through the application interface. The Interface sends the policy information, including the user identifier and vehicle details, to the Policy Service. The Policy Service then creates a new policy document and stores it in the MongoDB Atlas database before returning the generated policy identifier.

Next, the Application Interface initiates the payment process through the PaymentCubit by sending the policy identifier. The PaymentCubit communicates with the Payment Gateway to create a secure checkout session and receives a checkout URL, which is then returned to the User through the Interface in order to complete the payment.

An Alt combined fragment is used to represent the outcomes of the payment process:

• **[Payment successful]:** The Payment Gateway sends a checkout.paid webhook notification to the Webhook Controller. The system then updates the policy status to active in the database. After that, the PDF Service generates the insurance certificate PDF document containing the policy, vehicle, and user information. The generated file name and PDF link are returned to the system, while the NotificationCubit creates and stores a notification document if necessary. Finally, a real-time payment success notification containing the PDF link is emitted to the User through the Socket.io service.

• **[Else (payment canceled)]:** The Payment Gateway sends a checkout.canceled webhook notification to the Webhook Controller. In this case, the system deletes the corresponding policy document from the database to ensure data consistency.

3.7.4 Claim Submission and Assessment Process Sequence Diagram

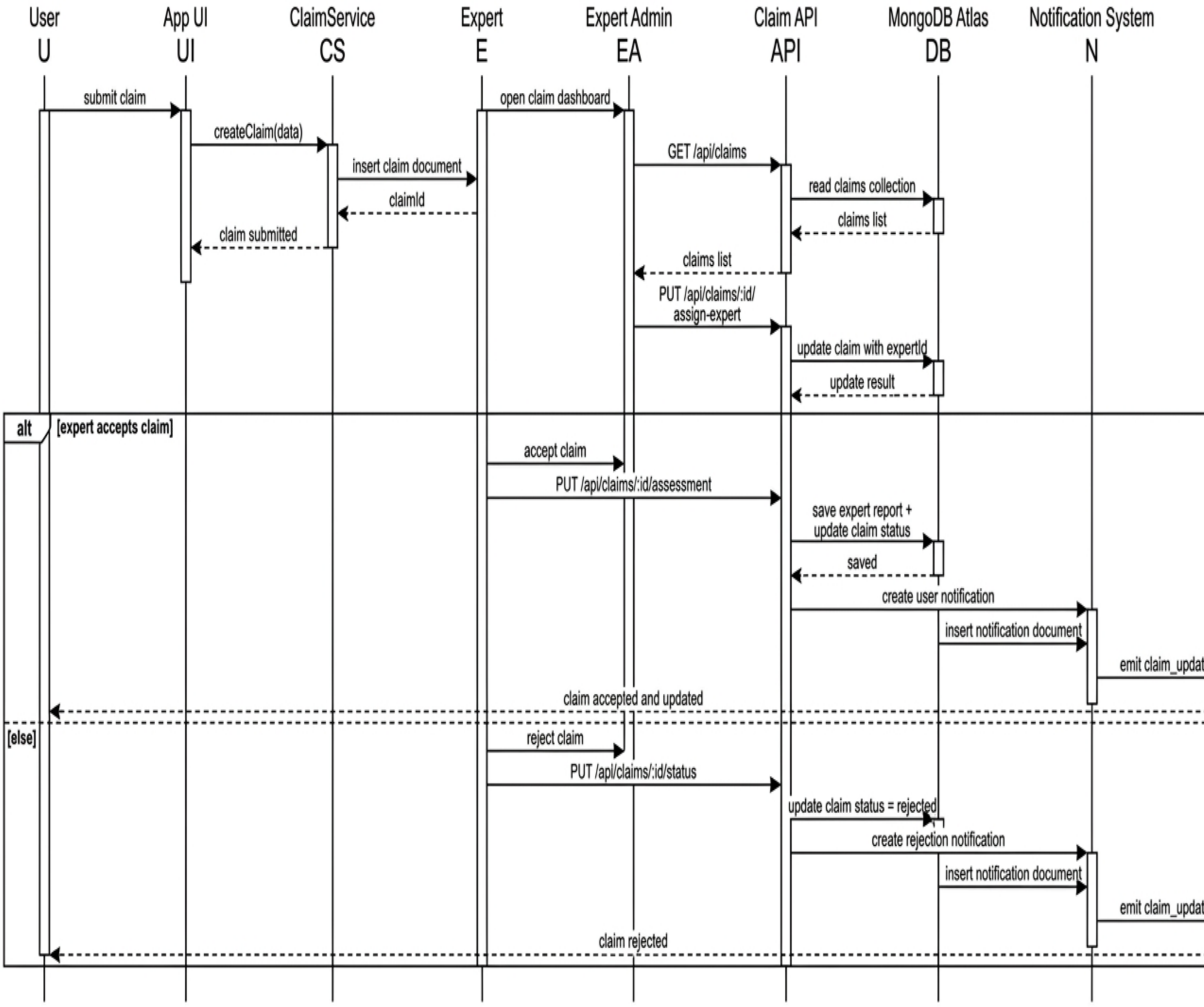


Figure 3.6 : Claim Submission and Assessment Process Sequence Diagram

Explanation:

The following sequence diagram models the interaction triggered when a user submits an insurance claim through the mobile application. The participants involved in this process are the User, the Application Interface, the ClaimService, the Expert, the Expert Admin, the Claim API, the MongoDB Atlas database, the Notification System, and the Socket.io service.

The process begins when the User submits a claim using the application interface. The Interface sends the claim data to the ClaimService, which creates a new claim document and stores it in the MongoDB Atlas database. After the claim is successfully inserted, the system returns a confirmation indicating that the claim has been submitted.

Next, the Expert opens the claim dashboard through the Expert Admin interface. The Expert Admin requests the list of claims from the Claim API, which retrieves the claims collection from the MongoDB Atlas database and returns the corresponding claims list. The Expert Admin then assigns an expert to the selected claim by sending an update request to the Claim API, which updates the claim document with the assigned expert information.

An Alt combined fragment is used to represent the possible outcomes of the claim assessment process:

• **[Expert accepts claim]:** The Expert accepts the claim and submits an assessment report through the Claim API. The system then saves the expert report and updates the claim status in the database. After that, the Notification System creates a user notification and stores the notification document in the database. Finally, a real-time claim update notification is emitted to the User through the Socket.io service, indicating that the claim has been accepted and updated.

• **[Else]:** If the claim is rejected, the system updates the claim status to rejected in the database. A rejection notification is then created and stored before being sent to the User through the Socket.io service, informing them that the claim has been rejected.

3.7.5 Identity Verification Sequence Diagram

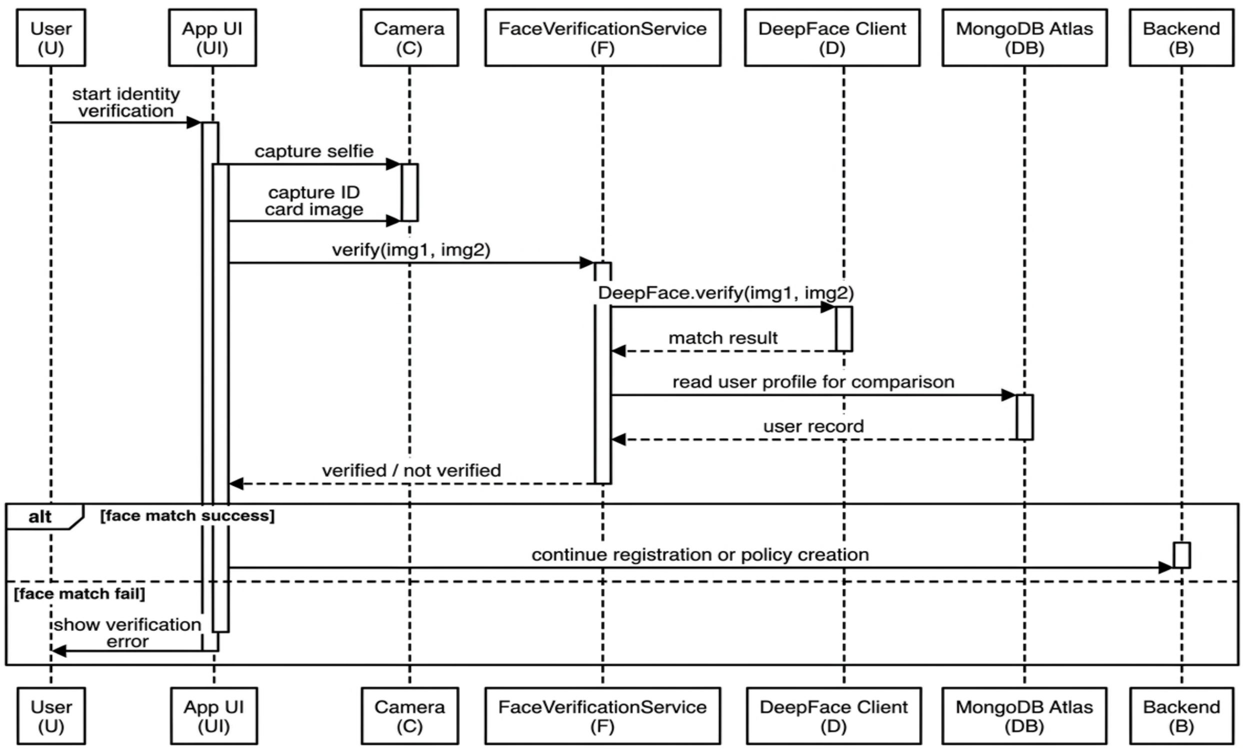


Figure 3.7 : Identity Verification Sequence Diagram

Explanation:

The following sequence diagram models the interaction triggered when a user performs identity verification through the mobile application. The participants involved in this process are the User, the Application Interface, the Camera, the FaceVerificationService, the DeepFace Client, the MongoDB Atlas database, and the Backend system.

The process begins when the User starts the identity verification procedure from the application interface. The Interface then activates the Camera in order to capture a selfie image and an image of the user's identification card. After both images are collected, the Application Interface sends them to the FaceVerificationService for verification.

Next, the FaceVerificationService communicates with the DeepFace Client, which performs facial comparison between the captured selfie and the identification card image using the DeepFace.verify(img1, img2) operation. The comparison result is then returned to the FaceVerificationService. In parallel, the service retrieves the user profile information from the MongoDB Atlas database in order to support the verification process.

An Alt combined fragment is used to represent the possible outcomes of the identity verification process:

• **[Face match success]:** If the facial comparison succeeds, the system confirms that the identity has been verified and allows the user to continue the registration process or policy creation workflow through the Backend system.

• **[Face match fail]:** If the facial comparison fails, the system returns a verification error message to the User through the application interface, indicating that the identity verification process was unsuccessful.

3.7.6 Claim Management and Expert Assessment Sequence Diagram

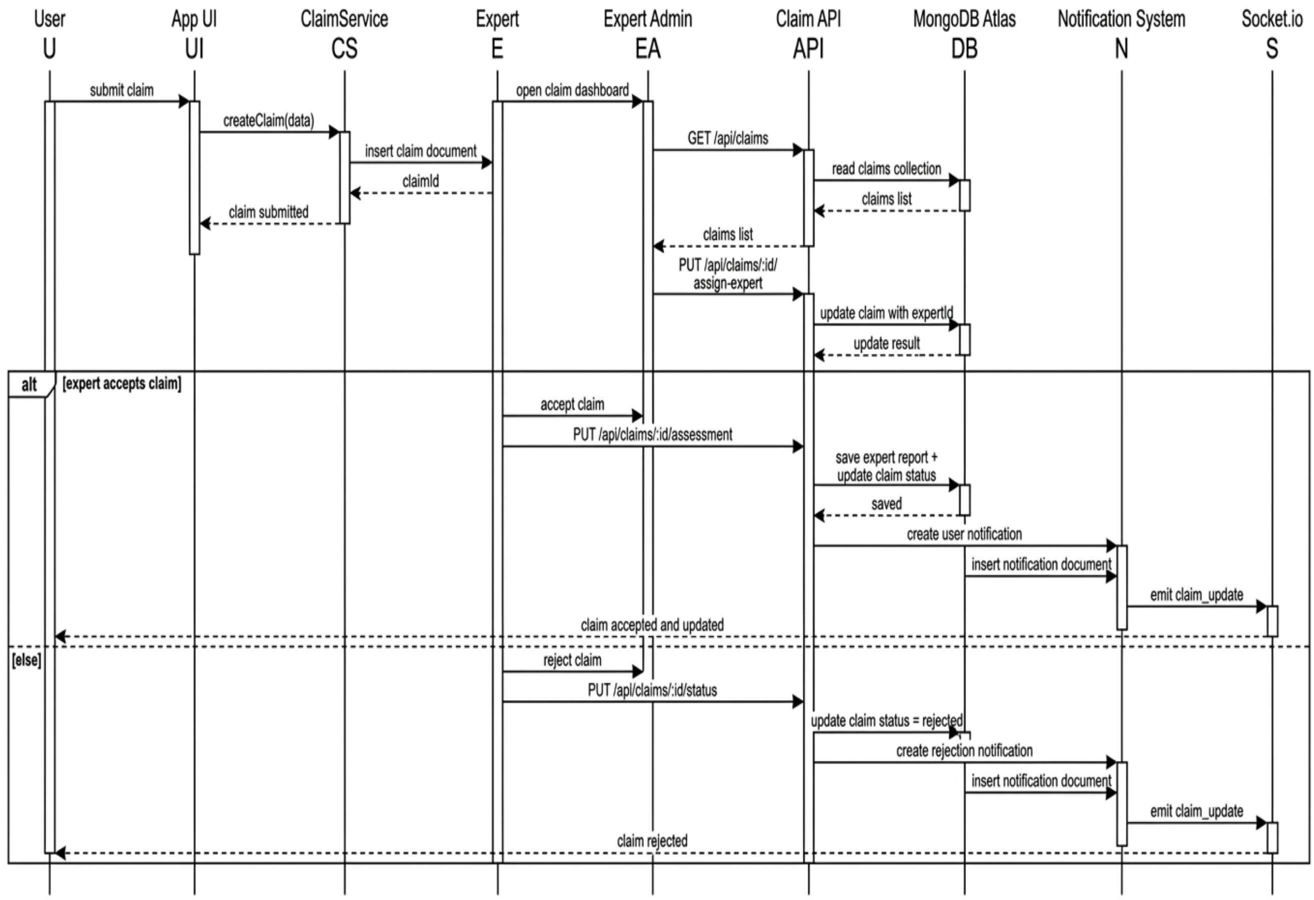


Figure 3.8 : Claim Management and Expert Assessment Sequence Diagram

Explanation:

The following sequence diagram models the interaction triggered when a user submits an insurance claim and when the expert evaluates the submitted claim. The participants involved in this process are the User, the Application Interface, the ClaimService, the Expert, the Expert Admin, the Claim API, the MongoDB Atlas database, the Notification System, and the Socket.io service.

The process begins when the User submits a claim through the application interface. The Interface sends the claim data to the ClaimService, which creates and inserts a new claim document into the MongoDB Atlas database. Once the claim is successfully stored, the system returns a confirmation message indicating that the claim has been submitted.

Next, the Expert opens the claim dashboard using the Expert Admin interface. The Expert Admin requests the list of claims from the Claim API, which retrieves the claims collection from the MongoDB Atlas database and returns the corresponding claims list. After reviewing the claims, the Expert Admin assigns an expert to a specific claim by sending an assignment request to the Claim API, which updates the claim document with the selected expert identifier.

An Alt combined fragment is used to represent the possible outcomes of the expert assessment process:

• **[Expert accepts claim]:** The Expert accepts the claim and submits an assessment report through the Claim API. The system then stores the expert report and updates the claim status in the MongoDB Atlas database. After that, the Notification System creates a notification for the user and stores the notification document in the database. Finally, a real-time claim update notification is emitted to the User through the Socket.io service, informing them that the claim has been accepted and updated successfully.

• **[Else]:** If the Expert rejects the claim, the system updates the claim status to rejected in the database. A rejection notification is then created and stored before being sent to the User through the Socket.io service, informing them that the claim has been rejected.

3.8 Conclusion

This chapter focused on the design and modelling of the proposed mobile insurance system using UML diagrams. Different modelling approaches were used to represent both the structural and behavioural aspects of the application.

The use case diagrams highlighted the interactions between the system actors and the principal services offered by the application, including user authentication, insurance subscription, claim declaration, payment operations, and identity verification. The class diagram provided an overview of the system structure by presenting the main entities and their relationships. Furthermore, the sequence diagrams illustrated the chronological exchange of messages between the different system components during the execution of major functionalities.

Overall, this design phase helped establish a clear vision of the system organisation and ensured that the application requirements were properly analysed before development. The next chapter will focus on the implementation phase and the technical realisation of the proposed solution.

4. Chapter 4 System Implementation and Technical Architecture

4.1 Introduction

Following the analysis and design phase presented in the previous chapter, this chapter focuses on the implementation and application of the proposed mobile insurance application. It explains how to translate the designed models and system architecture into a functional and interactive system.

The chapter begins by presenting the development environment, including the hardware and software tools used during the implementation process. It also introduces the main technologies and frameworks adopted for developing both the user interface and the server, as well as the external services integrated into the application.

Furthermore, this chapter describes the overall system architecture and explains the interaction between the different application layers. It also demonstrates the implementation of key functions, such as authentication, document management, claims processing, payment integration, and facial recognition, through numerous user interfaces and practical examples.

The chapter concludes with a testing and verification section, used to evaluate the proper functioning, reliability, and efficiency of the developed system.

4.2 Development Environment

4.2.1 Hardware Environment

The development and testing of the proposed system were carried out on a personal computer with the following hardware specifications:

Component	Specification
Processor	13th Gen Intel® Core™ i7-13620H @ 2.40 GHz
Installed RAM	16.00 GB (DDR4)
Storage	512 GB SSD

Component	Specification
Graphics Card	Intel® UHD Graphics

Table 4.1 : Hardware Specifications

4.2.2 Software Environment

The software tools and platforms used throughout the development process are summarized in the following table:

Software	Version	Role
Windows 10 Pro	22H2	Operating system used for development and testing.
Node.js	18+	JavaScript runtime for backend API and frontend build tools.
Python	3.8+	Primary programming language for face verification service.
Visual Studio Code	1.120.0+	Main code editor used throughout the development phase.
MongoDB	7.2.0	NoSQL database for storing user data, policies, claims.
Flutter SDK	>=3.5.0 <4.0.0	Mobile development framework for Android and iOS.
pnpm	8.0.0+	Package manager for frontend projects.

Software	Version	Role
Git	2.40+	Version control system for source code management.

Table 4.2 : Software Environment

4.3 Technologies and Tools

This section presents the main technologies and libraries used to develop the proposed full-stack vehicle insurance application, along with a justification for each choice.

4.3.1 Frontend Technologies

React 18.3.1 + TypeScript

was chosen as the UI framework for the administrative dashboard. React's component-based architecture allows for reusable and maintainable code, while TypeScript adds static type checking that significantly reduces runtime errors during development [18] [19].

Vite (v6.3.5)

was selected as the build tool due to its exceptionally fast development server and optimized production builds compared to traditional bundlers like Webpack [20].

TailwindCSS (v4.1.12)

provides a utility-first CSS approach that enables rapid and consistent UI development without leaving the HTML structure, ensuring a fully responsive design across all screen sizes [21].

Material-UI & Radix UI (v7.3.5 & v1.x)

offer a rich library of pre-built, accessible components that accelerate development and maintain a professional look across the dashboard interface [22].

Axios (v1.16.0)

was used as the HTTP client for all API communications between the frontend and the backend, offering a clean promise-based interface and built in request/response interceptors [23].

4.3.2 Backend Technologies

Node.js + Express.js (v18+ & v4.21.2)

These components form the core of a RESTful API server. The asynchronous, event-driven architecture of Node.js is ideal for efficiently handling multiple, simultaneous insurance policy requests and claims [24], [25].

MongoDB + Mongoose (v7.2.0 & v8.12.1)

The database layer is provided. MongoDB's flexible, document-oriented model adapts to the complex hierarchical nature of insurance data, from insurance documents and claims to user profiles, without requiring rigid relational diagrams [26]. Mongoose adds an ODM layer that enhances data validation and schema structure [27].

Socket.io (v4.8.1)

It enables real-time two-way communication between the server and connected clients, and is used for live notifications such as claims status updates and policy approvals [28].

JWT + bcryptjs (v9.0.2 & v2.4.3)

It handles authentication and security. JWT tokens manages stateless session authentication across mobile and web clients, while bcryptjs ensures that user passwords are not stored as plain text [29] [30].

PDFKit (v0.17.1)

is used to generate insurance policy documents and claim reports programmatically as downloadable PDF files directly from the backend [31].

Multer (v1.4.5-lts.1)

serves as the file upload middleware, handling the submission of accident photos, identity documents, and vehicle registration files from both the mobile app and the dashboard [32].

4.3.3 Mobile Technologies

Flutter + Dart (>=3.5.0 <4.0.0)

Flutter was chosen for developing cross-platform mobile applications, enabling the use of a single codebase to target both Android and iOS. Flutter's rich library of tools ensures a consistent and seamless user experience on both systems [33].

Camera (v0.11.3) & Image Picker (v1.2.0)

provide access to the device camera and gallery, enabling users to capture and upload accident scene photos and identity documents directly from the mobile app [34], [35].

Flutter Secure Storage (v9.2.2)

ensures that sensitive user credentials and authentication tokens are stored in encrypted form on the device, protecting against unauthorized access [33].

WebView Flutter (v4.13.0)

allows embedded web content rendering within the mobile app, used for displaying policy documents and payment gateway pages [33].

4.3.4 Face Verification Service

Flask (v3.1.0)

was chosen as the Python web framework for the face verification microservice due to its lightweight nature and minimal overhead, making it ideal for a focused single-purpose service [36].

DeepFace (v0.0.75)

provides the core face recognition and verification capabilities. It supports multiple deep learning model backends and delivers high accuracy in matching facial images against stored identity references [37].

Keras (v2.10.0) & OpenCV (v4.11.0.86)

support the deep learning inference pipeline and image preprocessing respectively. OpenCV handles face detection, image resizing, and format normalization before images are passed to the DeepFace engine [38].

Gunicorn (v23.0.0)

serves as the production WSGI server, enabling the Flask microservice to handle multiple concurrent verification requests efficiently [39].

Pandas (v2.2.3) & NumPy (v1.24.4)

support data processing and numerical computing tasks within the verification pipeline [40], [41].

4.3.5 Supporting Libraries and Tools

Library/Tool	Version	Role
Dio	5.9.0	HTTP client for Flutter
Chargily Pay	2.1.0	Payment gateway integration
Git	2.40+	Version control
Postman	11.0.0+	API testing tool
Nodemon	3.1.9	Node.js auto-restart development tool

Table 4.3 : Supporting Libraries and Tools

4.4 System Architecture

The vehicle insurance application is structured into four distinct layers, with each layer serving a specific purpose within the system. As shown in Figure 4.1, these layers communicate with one another, enabling smooth data flow throughout the application.

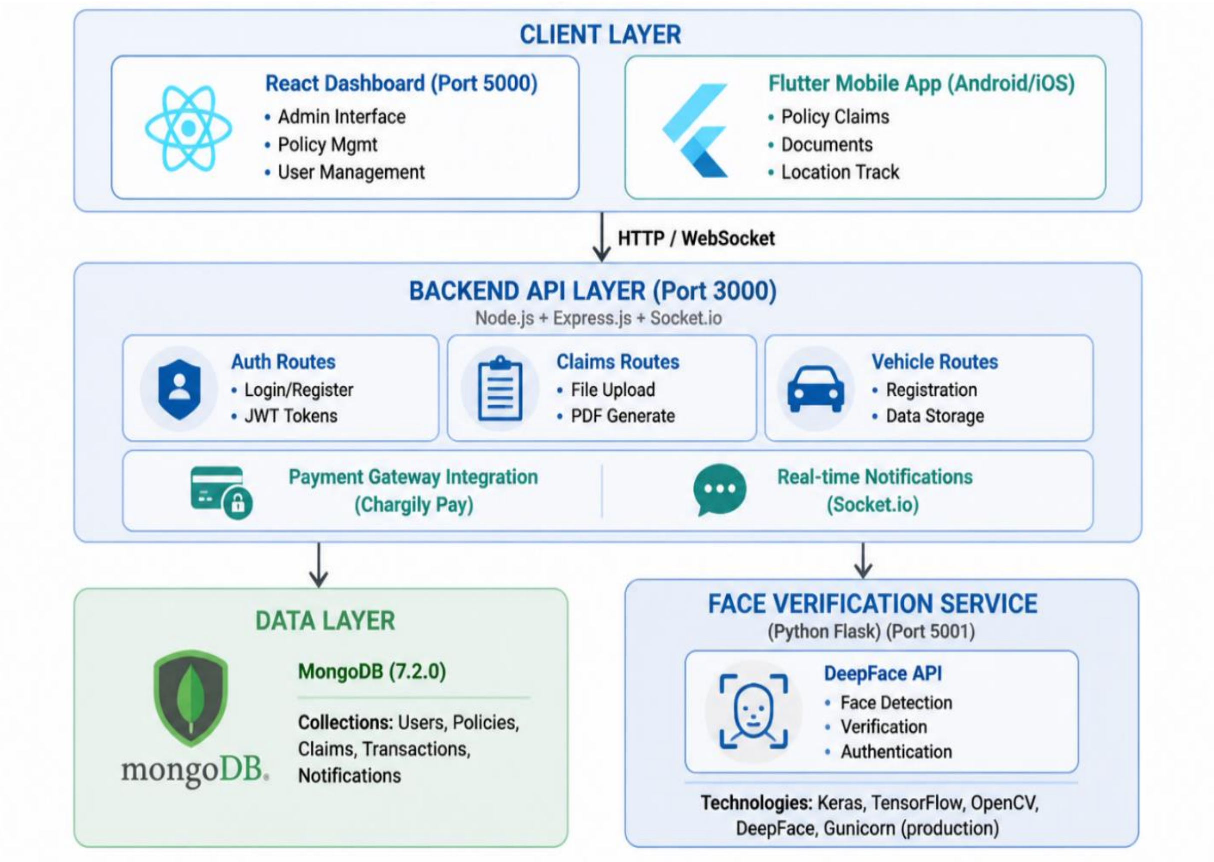


Figure 4.1 : System Architecture

The Client Layer includes a React Admin Dashboard for system management and a Flutter mobile application for customers to manage policies, submit claims, and upload documents. Both applications communicate with the backend using HTTP and WebSocket protocols.

The Backend API Layer is developed using Node.js, Express.js, and Socket.io. It handles business logic, authentication, claims management, vehicle management, payment integration, and real-time notifications.

Communication between the client applications and the backend server is achieved through RESTful APIs over HTTP/HTTPS protocols. JWT tokens are used to secure authenticated requests and manage user sessions.

The Data Layer uses MongoDB with Mongoose ODM to store users, policies, claims, transactions, and notifications. This NoSQL database provides flexibility and scalability for the insurance system.

The Face Verification Service operates as an independent microservice using Python Flask and DeepFace technologies. It performs biometric identity verification through facial recognition to enhance system security.

Overall, this architecture ensures modularity, scalability, security, and maintainability while facilitating future system expansion.

4.5 Database Design

The MongoDB database (version 7.2.0) was used in developing the vehicle insurance application due to its high flexibility, ease of expansion, and ability to efficiently handle data similar to JSON documents [26]. Unlike relational databases, MongoDB's structure allows for the storage of complex and nested data within each record, which is suitable for insurance data needs that include documents, claims, and user files with diverse hierarchical structures. To ensure data integrity and ease of management, the Mongoose library (version 8.12.1) was employed as an ODM (Open Data Management) layer, providing precise control over schemas and facilitating various database operations [27].

The database was organized into seven collections, each responsible for storing a specific type of application data.

4.5.1 Users Collection

The Users collection stores all registered user accounts, including both customers and administrators. Each document contains the following fields:

Field	Type	Description
_id	ObjectId	Unique user identifier
fullName	String	User's full name
email	String	Email address (unique)
password	String	Encrypted password
phone	String	Phone number
address	String	Physical address
specialty	String	Specialization (for experts)
roles	Map	User roles (User: 2001, Expert: 2002)
refreshToken	String	Session refresh token
createdAt	Date	Account creation date

Table 4.4 : Users Collection

The Explanation:

The roles field determines the level of access granted to each user across both the mobile application and the administrative dashboard, ensuring that experts can access claim assessments while regular users are limited to policy and claim submission. The refreshToken field stores the session token used to maintain secure authentication, allowing users to stay logged in across devices without repeatedly entering credentials. The specialty field is reserved for expert users, identifying their area of specialization (e.g., automotive damage assessment), which helps the system assign claims to the most qualified assessor.

4.5.2 Policies Collection

The Policies collection stores all insurance policy records created and managed through the system. Each document contains the following fields:

Field	Type	Description
_id	ObjectId	Unique policy identifier
user	ObjectId	Reference to User
vehicle	ObjectId	Reference to Vehicle
type	String	Policy type
startDate	String	Policy starts date
endDate	String	Policy end date
price	Number	Premium amount
status	Enum	pending, active, expired, cancelled
paymentStatus	Enum	pending, paid, failed
checkoutId	String	Stripe payment ID
renewalOf	ObjectId	Original policy ID (renewals)
isRenewal	Boolean	Renewal flag
notified	Boolean	Expiration notification sent
createdAt	Date	Creation date

Table 4.5 : Policies Collection

The Explanation:

The status field tracks the lifecycle of an insurance policy, ensuring transitions from pending to active, and later to expired or cancelled. The checkoutId field stores the Stripe payment session identifier, linking each policy to its corresponding financial transaction.

4.5.3 Claims Collection

The Claims collection records all accident claims submitted by customers through the mobile application. Each document contains the following fields:

Field	Type	Description
_id	ObjectId	Unique claim identifier
user	ObjectId	Reference to User (claimant)
vehicle	ObjectId	Reference to Vehicle
policy	ObjectId	Reference to Policy
expert	ObjectId	Reference to Expert
description	String	Accident/incident description
date	Date	Date of incident
location	String	Incident location
wilaya	String	Province/Region
accidentType	Enum	collision, theft, fire, vandalism, natural disaster, other
VehicleStatus	[String]	Vehicle damage descriptions
images	[String]	Claim incident images

Field	Type	Description
status	Enum	pending, in_review, resolved, rejected
assessment	Object	Expert assessment data

Table 4.6 : Claims Collection

The Explanation:

The accidentType field categorizes the nature of the incident (collision, theft, fire, vandalism, natural disaster, other), which is essential for determining claim eligibility and applying the correct processing rules. The assessment field contains expert evaluation data, including repair cost estimates, supporting photos, and insurance price recommendations, forming the basis for claim resolution and payout decisions. The VehicleStatus array documents specific damages to the vehicle, ensuring transparency and accuracy in the repair process.

4.5.4 Transactions Collection

The Transactions collection records all payment operations processed through the Chargily Pay gateway. Each document contains the following fields:

Field	Location	Type	Description
checkoutId	**Policy**	**String**	**Chargily payment session ID**
paymentStatus	**Policy**	**Enum**	**pending, paid, failed**
price	**Policy**	**Number**	**Transaction amount**
createdAt	**Policy**	**Date**	**Transaction timestamp**

Table 4.7 : Transactions Collection

The Explanation:

The paymentStatus field reflects the outcome of the payment process (pending, paid, failed), directly impacting whether a policy becomes active. The createdAt field serves as the transaction timestamp, ensuring accurate financial tracking and audit compliance.

4.5.5 Notifications Collection

The Notifications collection stores all system-generated alerts sent to users in response to claim updates, policy activations, and payment confirmations. Each document contains the following fields:

Field	Type	Description
_id	**ObjectId**	**Unique notification identifier**
user	**ObjectId**	**Reference to recipient User**
claim	**ObjectId**	**Reference to related Claim**
expert	**ObjectId**	**Reference to related Expert**
policy	**ObjectId**	**Reference to related Policy**
vehicle	**ObjectId**	**Reference to related Vehicle**
title	**String**	**Notification title**
message	**String**	**Notification message**
type	**Enum**	**claim_approved, claim_rejected, claim_pending, claim_in_review, payment_received, renewal_success, expert_report_ready**
pdfUrl	**String**	**URL to attached PDF**
misreads	**Boolean**	**Read status**

Field	Type	Description
createdAt	**Date**	**Creation timestamp**
updatedAt	**Date**	**Last update timestamp**

Table 4.8 : Notifications Collection

The Explanation:

The type field specifies the trigger event (e.g., claim approved, payment received, expert report ready), allowing the system to deliver context-specific alerts to users and keep them informed of important updates. The pdfUrl field stores the path to downloadable reports or certificates, ensuring users can access official documentation linked to their claims or policies. The isRead field tracks whether the user has opened the notification, enabling the system to manage unread alerts and improve user engagement.

4.5.6 ExpertReport Collection

The ExpertReport collection stores inspection reports created by experts for submitted claims. Each report includes details of vehicle damage, estimated repair costs, supporting photos, and a generated PDF document. Key fields are:

Field	Type	Description
_id	**ObjectId**	**Unique report identifier**
claim	**ObjectId**	**Reference to Claim**
expert	**ObjectId**	**Reference to Expert User**
inspectionDate	**ISODate**	**Date of inspection**
damageDescription	**String**	**Detailed damage assessment**
estimatedCost	**Number**	**Estimated repair cost**
images	**[String]**	**Assessment photos**

Field	Type	Description
pdfUrl	String	Generated PDF report URL
createdAt	ISODate	Creation date
updatedAt	ISODate	Last update date

Table 4.9 : ExpertReport Collection

The Explanation:

The damageDescription field provides a detailed narrative of the vehicle's condition after inspection, supporting transparency in claim assessments. The pdfUrl field stores the generated expert report in PDF format, which serves as an official record for both the claimant and the insurance company.

4.5.7 Vehicle Collection

The Vehicle collection stores essential details about insured vehicles and links each record to its owner. It ensures that every vehicle is uniquely identified and properly documented for insurance operations. Each document contains the following fields:

Field	Type	Description
_id	ObjectId	Unique vehicle identifier
owner	ObjectId	Reference to User
registrationNumber	String	License plate
model	String	Vehicle model
brand	String	Vehicle brand/make
vehicleType	String	sedan, suv, truck
usageType	String	personal, commercial

Field	Type	Description
year	**Number**	**Manufacturing year**
chassisNumber	**String**	**VIN**
driveLicense	**String**	**License document URL**
vehicleRegistration	**String**	**Registration document URL**
claims	**Number**	**Total claims count**

Table 4.10 : Vehicle Collection

The Explanation:

The registrationNumber field uniquely identifies the vehicle within the system, ensuring accurate mapping between policies, claims, and ownership records. The claims field tracks the total number of claims associated with the vehicle, which can influence risk assessment, premium calculations, and fraud detection. The chassisNumber field (VIN) provides a globally unique identifier for the vehicle, ensuring that duplicate or fraudulent registrations are prevented.

4.6 Implementation of Main Functionalities

This section presents the practical realization of each core functionality of the system. For each module, the implementation logic is described alongside the corresponding code listings and user interface screenshots.

The system runs across four parallel runtime components: the Node.js backend API server on port 3000, the React administrative dashboard on port 5000, the Flutter mobile application on Android and iOS devices, and the Flask face verification microservice on port 5001.

4.6.1 Authentication

The authentication module of the **Carity** application covers two distinct processes: **user registration** and **user login**. The registration process guides new customers through a structured four-step onboarding flow, while the login process allows returning users to securely access their accounts using their registered email address and password, authenticated through a JWT-based token mechanism.

Registration Process

Unlike traditional one-step registration models, the Caree app employs a guided four-step registration process that gathers all the necessary information before creating the user account and its associated insurance policy. To gain full access to the app, the customer must complete the following four steps:

- **Profile:**

The customer begins by filling in their personal information including full name, email address, phone number, and home address. This information is stored in the Users collection in MongoDB and forms the base of the user account.

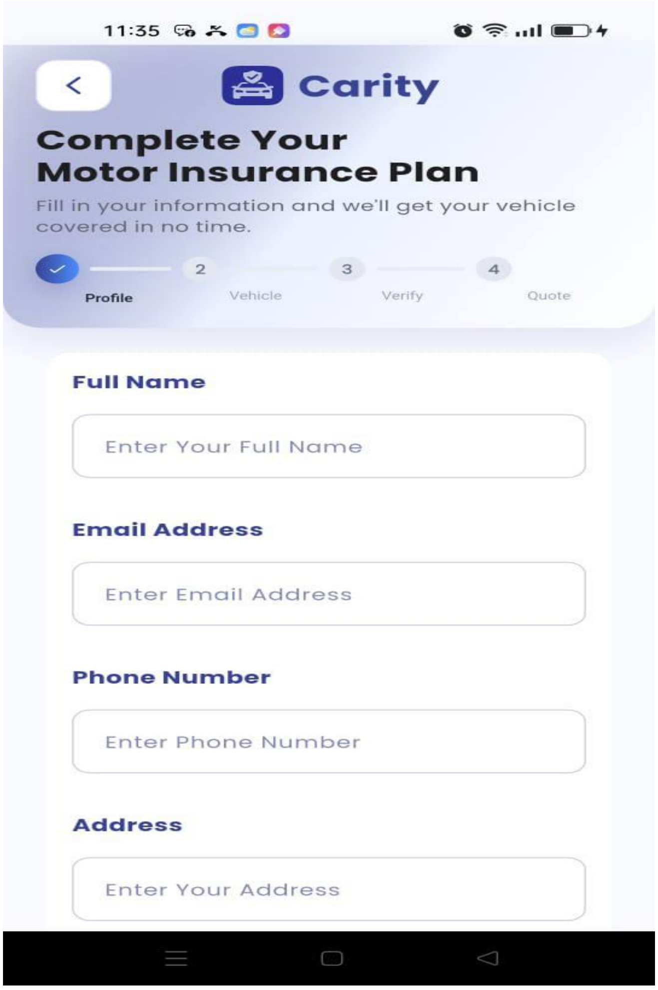


Figure 4.2 : Profile Information Form, Registration Process

- **Vehicle:**

The customer then provides their vehicle details including vehicle type, model, usage type, and registration number. The vehicle type must be selected first, after which the available models are dynamically loaded.

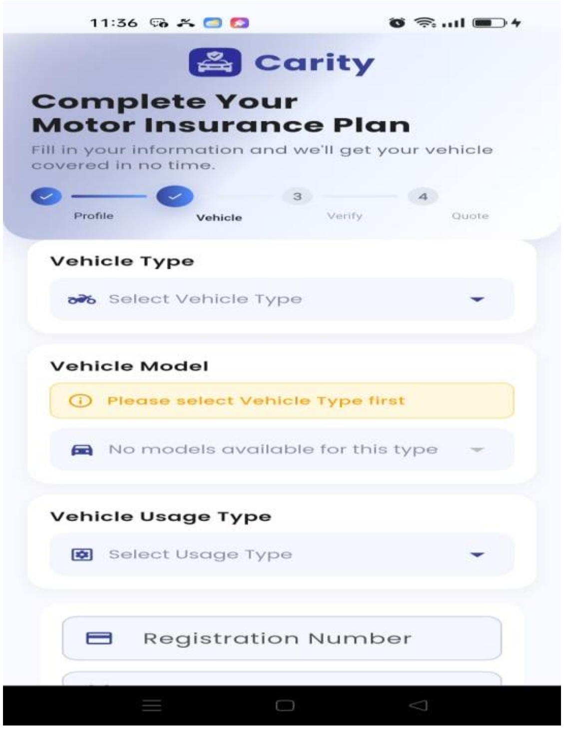


Figure 4.3 : Vehicle Details Form, Registration Process

- **Verify:**

The customer is prompted to capture a clear selfie for biometric face verification against the driver's license photo submitted in Step 2. The captured image is transmitted to the Flask face verification microservice for identity confirmation.

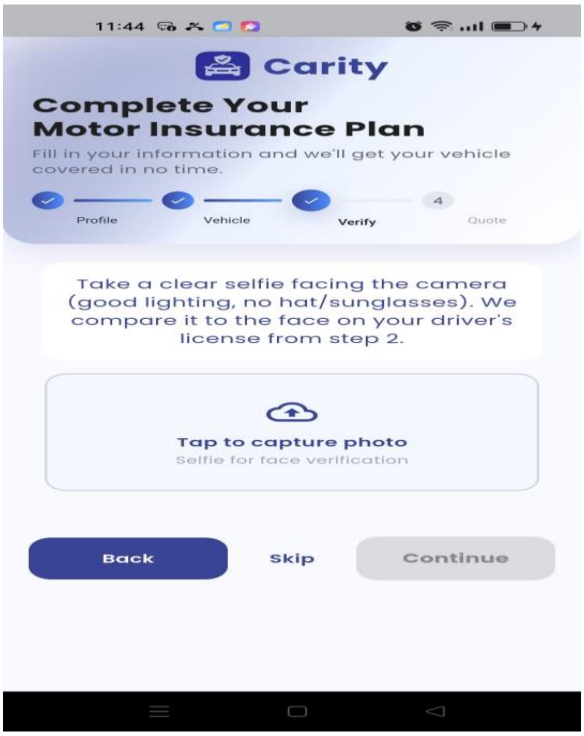


Figure 4.4 : Face Verification Screen, Registration Process

- **Quote:**

Finally, the customer browses available insurance plans filtered by coverage type and selects the plan that best suits their needs. Each plan displays the coverage type, validity period, included benefits, and the premium amount in DZD. This step is detailed further in Section 4.6.2.

Login Flow

Once registration is complete, returning users access the application through the standard login screen by providing their email address and password. The backend retrieves the corresponding account from the Users collection, compares the submitted password against the stored bcrypt hash using the bcrypt.compare() function, and upon successful validation issues two tokens: a short-lived **access token** valid for 7200 seconds and a long-lived **refresh token** valid for one day. The refresh token is persisted in the user's database document to support session renewal without requiring re-authentication. All protected API routes are guarded by a JWT verification middleware that validates the token's signature and expiry on every incoming request, returning a 401 Unauthorized response for invalid or expired sessions.

The following code excerpt presents the implementation of the authentication controller responsible for validating user credentials and generating secure JWT access tokens:

```
const match = await bcrypt.compare(pwd, foundUser.password);
if (match) {
  const roles = Object.values(foundUser.roles);
  const accessToken = jwt.sign(
    {
      UserInfo: {
        email: foundUser.email,
        roles: roles,
      },
    },
    process.env.JWT_SECRET,
    { expiresIn: "7200s" }
  );
  const refreshToken = jwt.sign(
    { email: foundUser.email },
    process.env.REFRESH_TOKEN_SECRET,
    { expiresIn: "1d" }
  );
  res.json({
    success: true,
    accessToken,
    refreshToken,
  });}
```

Figure 4.5 : JWT-Based Authentication Controller

The administrative dashboard provides a dedicated login interface for expert accounts accessible at localhost:5173/login. The expert enters their registered email address and password to sign in through the Car Assurance Expert — Claims Assessment Platform interface.

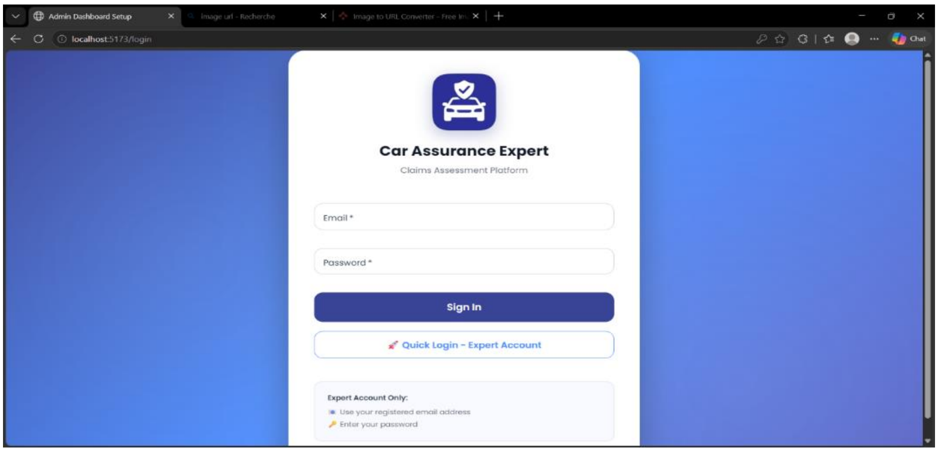


Figure 4.6 : Login Interface Administrative Dashboard (Web)

After successful login, the customer is redirected to the Home Screen, which displays a promotional banner, a Profile Completion progress indicator, and a bottom navigation bar providing quick access to the Home, Claims, and Notifications sections.

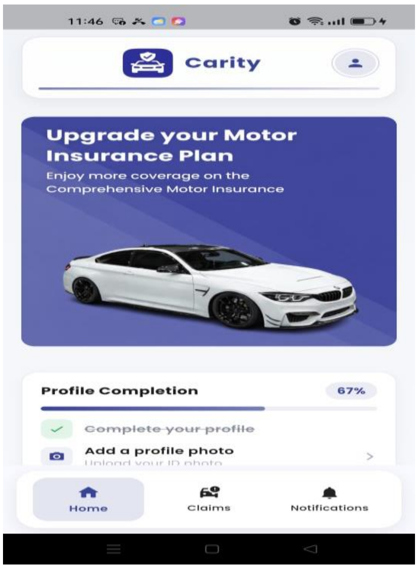


Figure 4.7 : Home Screen Main Dashboard

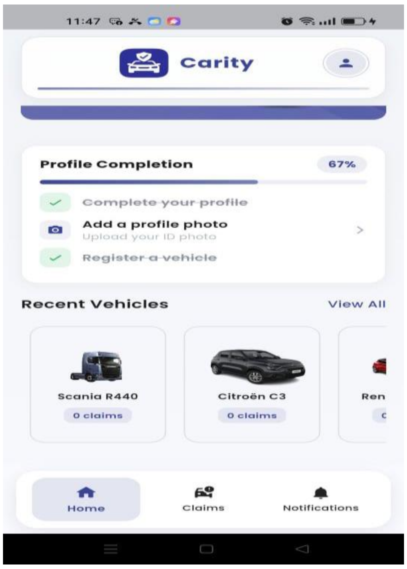


Figure 4.8 : Home Screen Profile Completion and Recent Vehicles Section

The customer can also access their personal profile through the profile icon located in the top right corner of the home screen. The profile screen displays the user's account information including full name, email address, and profile photo, and allows the customer to update their personal details.

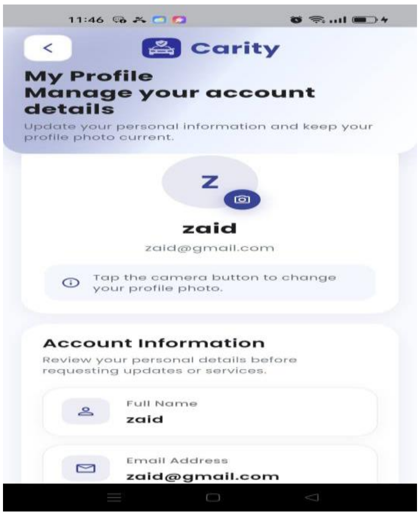


Figure 4.9 : My Profile Account Details Screen

4.6.2 Policy Management

The policy management module allows customers to browse available insurance plans and select the coverage that best suits their needs. After completing the onboarding steps, the customer reaches Step 4: Quote, where available plans are displayed with their coverage type, validity period, and premium amount in DZD. Upon selecting a plan, the customer confirms the coverage dates on the Create Policy screen, where the system automatically calculates the policy start date and end date based on the selected plan duration.

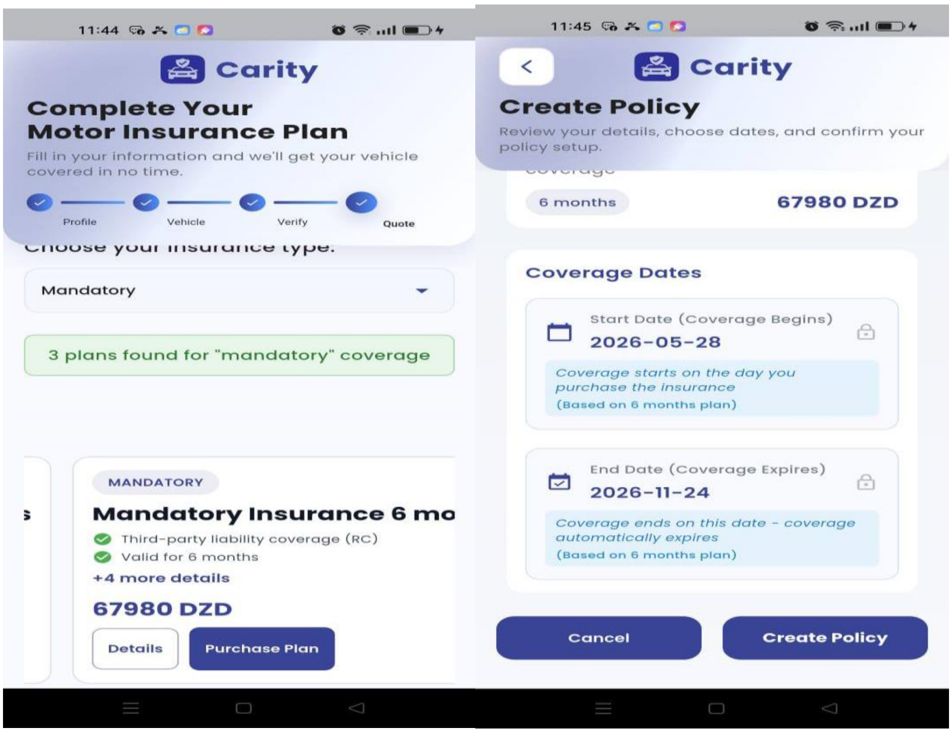


Figure 4.10 : Insurance Quote Selection and Coverage Dates Confirmation Screens

Motor Insurance Certificate

Coverage period: 5/25/2026 — 8/23/2026

Policy Information

Policy ID: 6a141c26517b33fe2bbd2f37
Issue Date: 5/25/2026
Start Date: 5/25/2026
End Date: 8/23/2026
Payment Status: paid

Insured Person

Full Name: zaid
Email: zaid@gmail.com
Phone: 0672399079
Address: Google Building 40, Mountain View, California, United States

Vehicle Details

Type: Truck
Brand: N/A
Model: Volvo Trucks FH16
Year: 2021
Registration Number: 123455
Usage Type: Commercial
Chassis Number: 29392943
Vehicle Registration Doc: On file

Coverage Details

Plan / Policy Type: mandatory
Premium Amount: 21800 DZD
Status: Active
Coverage: Comprehensive

This certificate was generated on 5/25/2026 at 10:56:59 AM
Generated automatically by Insurance Management System

Figure 4.11 : Motor Insurance Certificate

Upon confirmation, the backend stores the policy in MongoDB and automatically generates a Motor Insurance Certificate PDF using PDFKit, containing all policy details including the policy ID, coverage period, insured person information, vehicle details, and coverage type. This certificate is delivered to the customer and made available for download within the mobile application.

4.6.3 Claims Management

The claims management module enables customers to report traffic accidents and submit insurance claims through the Flutter mobile application. The claim submission process is divided into **two sequential steps**:

- **Accident Details:**

The customer selects the involved vehicle from their registered vehicle list, specifies the accident type, and enters the accident location including the wilaya. The GPS coordinates of the incident are recorded automatically via the Geolocator package, providing precise location data for the claim record.

Figure 4.12 : Report Traffic Accident Details

- **Accident Description:**

The customer provides a detailed written description of the accident, including information about weather conditions, visibility, damages, witnesses, and the sequence of events. Once the description is complete, the customer taps Confirm and Complete to submit the claim. The submitted claim is stored in the Claims collection in MongoDB, and the administrator is notified in real time via Socket.io.

Figure 4.13 : Report Traffic Accident Description

Administrators review all submitted claims through the Claims Assessment screen of the administrative dashboard, where each claim displays the vehicle name, claimant username, accident location, submission date, and current status badge Approved, Rejected, or Pending. Each status transition triggers a real-time Socket.io notification delivered instantly to the customer's mobile application.

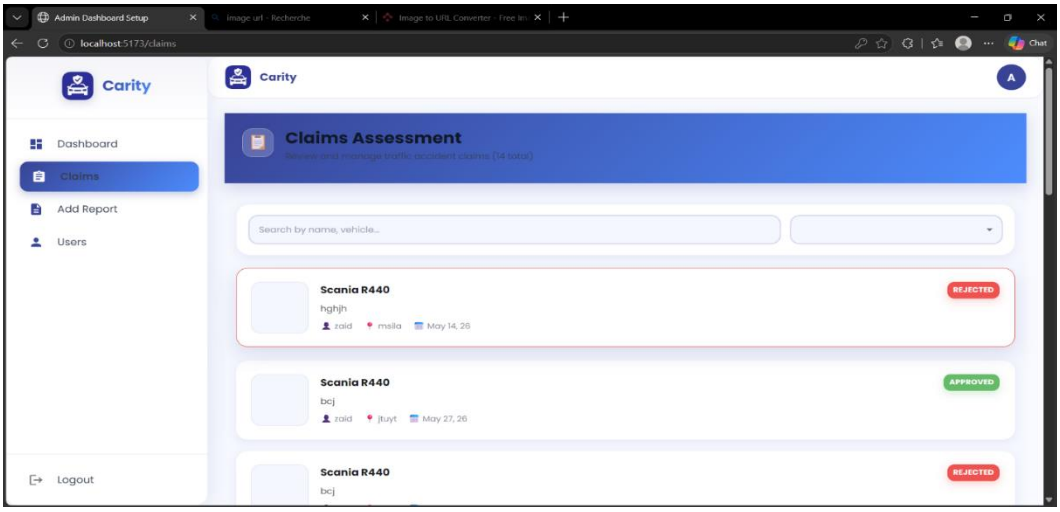
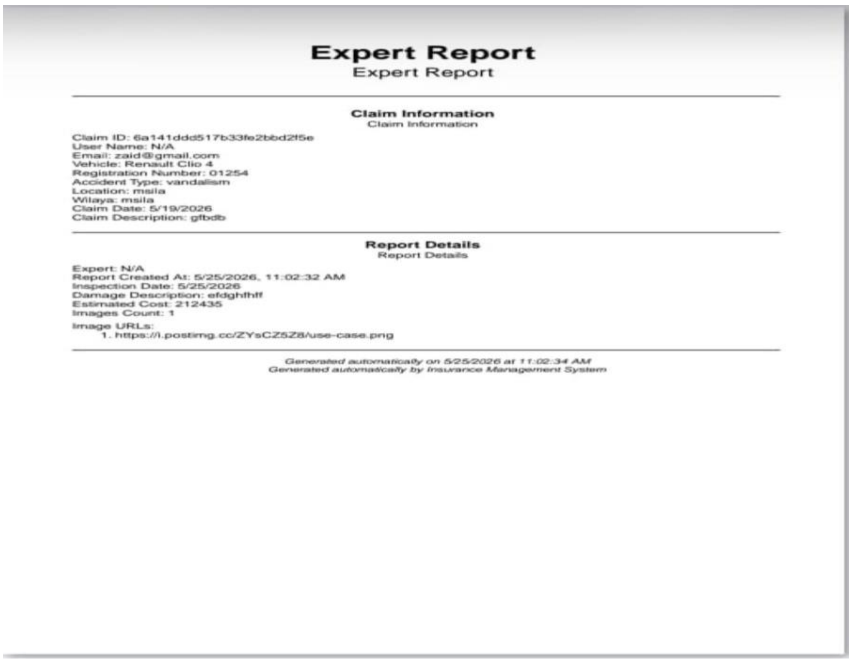


Figure 4.14 : Claims Management Screen — Administrative Dashboard

Once the assessment is complete, the expert submits a detailed Expert Report generated automatically as a PDF document containing the claim information, vehicle details, accident type, damage description, estimated cost, and supporting image references. This report is delivered to the customer via a real-time Socket.io notification.



Expert Report
Expert Report

Claim Information
Claim Information

Claim ID: 6a141ddd517b33fe2bbd2f5e
User Name: N/A
Email: zaid@gmail.com
Vehicle: Renault Clio 4
Registration Number: 01254
Accident Type: vandalism
Location: msila
Wilaya: msila
Claim Date: 5/19/2026
Claim Description: gfbdb

Report Details
Report Details

Expert: N/A
Report Created At: 5/25/2026, 11:02:32 AM
Inspection Date: 5/25/2026
Damage Description: efdghfhff
Estimated Cost: 212435
Images Count: 1
Image URLs:
1. https://i.postimg.cc/ZYsCZ5Z8/use-case.png

Generated automatically on 5/25/2026 at 11:02:34 AM
Generated automatically by Insurance Management System

Figure 4.15 : Expert Report

4.6.4 Vehicle Registration

The vehicle registration module allows customers to register their vehicles either as part of the **four-step onboarding flow** or independently through the **Add New Vehicle** screen, by providing the vehicle type, model, usage type, registration number, manufacturing year, and chassis number, which are then stored in the Vehicle collection in MongoDB.

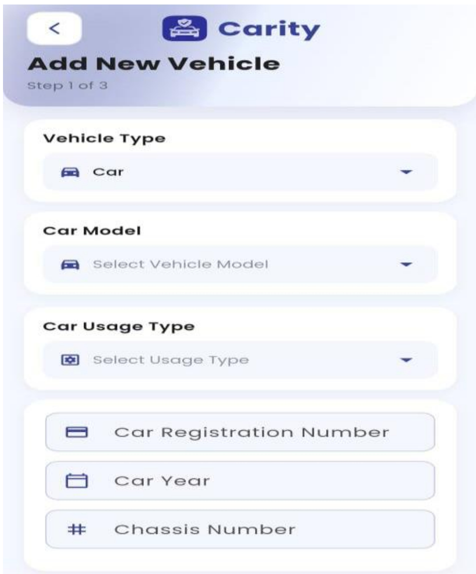


Figure 4.16 : Add New Vehicle Details Form

The customer can also view all their registered vehicles through the Your Vehicles screen, which displays each vehicle's details, current insurance status, remaining coverage days, and a Renew Now button for policies approaching expiry.

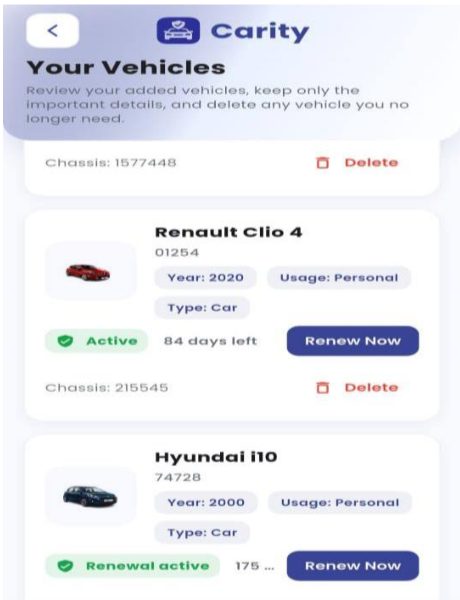


Figure 4.17 : Registered Vehicles List Screen

4.6.5 Payment Integration

Payment processing is handled through the Chargily Pay gateway. After policy confirmation, the backend generates a secure payment session rendered inside a WebView widget, where the customer selects their preferred payment method EDAHABIA Card, CIB Card, or Chargily App. Upon successful payment, the policy is activated and a real-time notification is sent to the user via Socket.io.

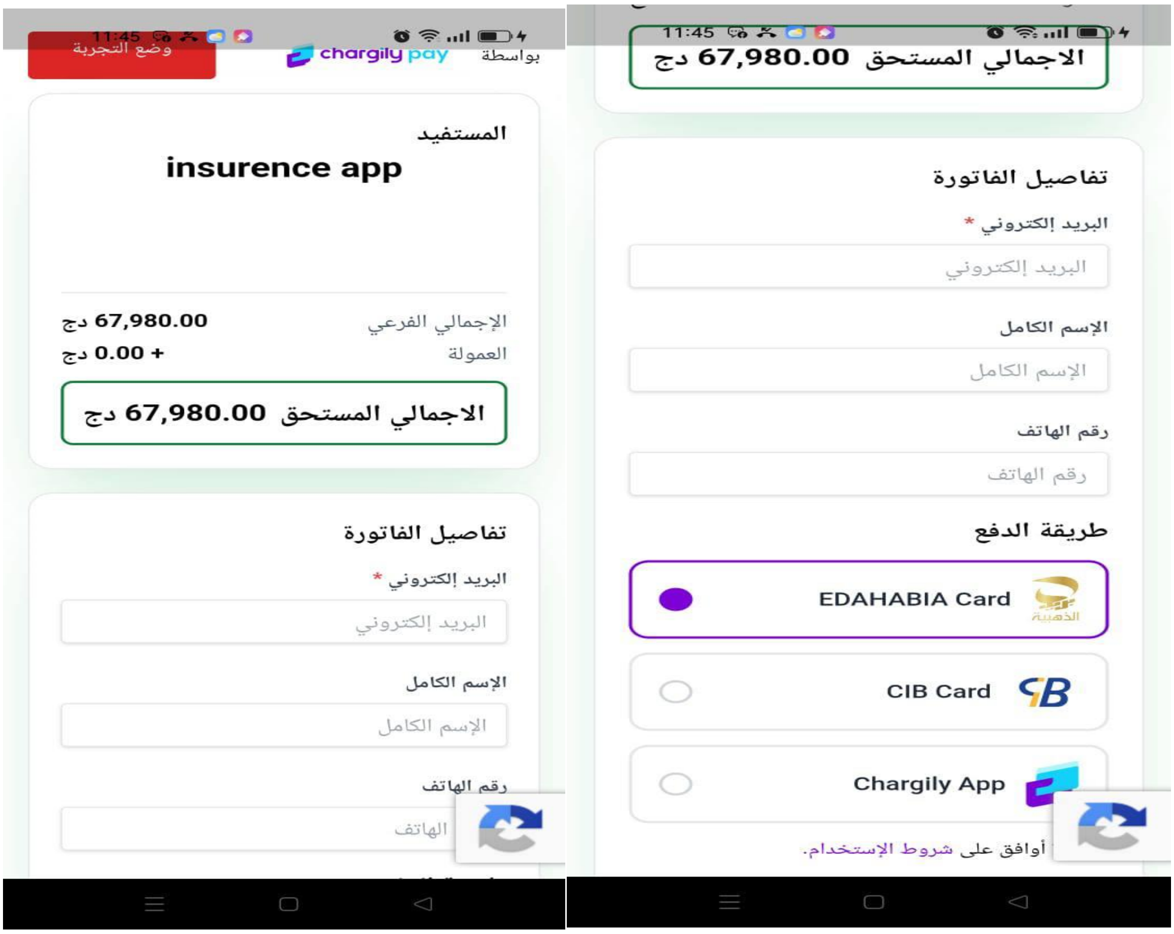


Figure 4.18 : Chargily Pay Billing Details and Payment Method Selection

4.6.6 Face Verification Service

The face verification microservice is integrated as **Step 3: Verify** within the onboarding flow. After completing the vehicle registration step, the customer is prompted to capture a clear selfie which is transmitted to the Flask microservice operating independently on port 5001. OpenCV preprocesses the received image by performing resizing, color normalization, and face region detection, before passing it to the DeepFace engine for comparison against the reference identity photo stored during registration. The service returns a JSON response containing the verification status and a confidence score indicating the degree of facial similarity.

```javascript
const match = await bcrypt.compare(pwd, foundUser.password);
if (match) {
  const roles = Object.values(foundUser.roles);
  const accessToken = jwt.sign(
    {
      UserInfo: {
        email: foundUser.email,
        roles: roles,
      },
    },
    process.env.JWT_SECRET,
    { expiresIn: "7200s" }
  );
  const refreshToken = jwt.sign(
    { email: foundUser.email },
    process.env.REFRESH_TOKEN_SECRET,
    { expiresIn: "1d" }
  );
  res.json({
    success: true,
    accessToken,
    refreshToken,
  });}
```

Figure 4.19 : DeepFace-Based Identity Verification Controller

4.6.7 Real-Time Notifications

Real-time communication is implemented using **Socket.io**, which emits targeted events to specific users whenever a significant state change occurs — including claim status updates, policy approvals, and payment confirmations. These events are rendered in the dedicated **Notifications** screen, where customers can view updates, open expert reports, and receive their insurance certificate notifications.

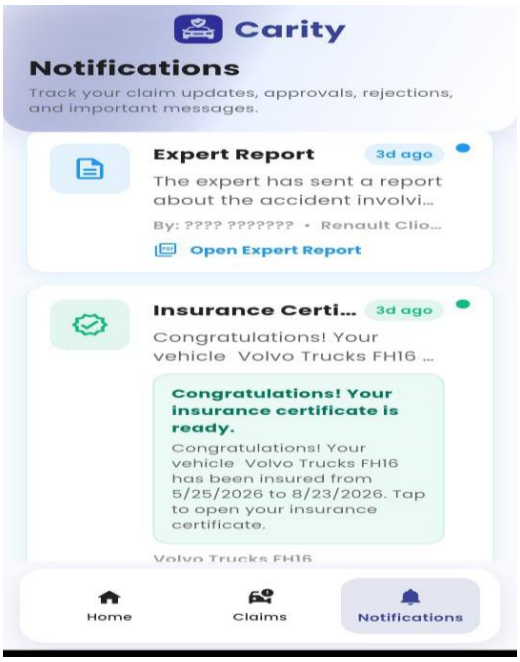


Figure 4.20 : Notifications Screen Real-Time Alerts and Updates

4.6.8 Administrative Dashboard

In addition to the mobile application, the system includes a web-based Administrative Dashboard accessible to expert accounts, built using React 18.3.1 with TypeScript and served on port 5173 [18]. The sidebar navigation provides access to four main sections: Dashboard, Claims, Add Report, and Users. The main dashboard screen provides a real-time overview of key statistics including total claims, pending claims, and active users, alongside a Claims by Status breakdown and a Recent Claims list.

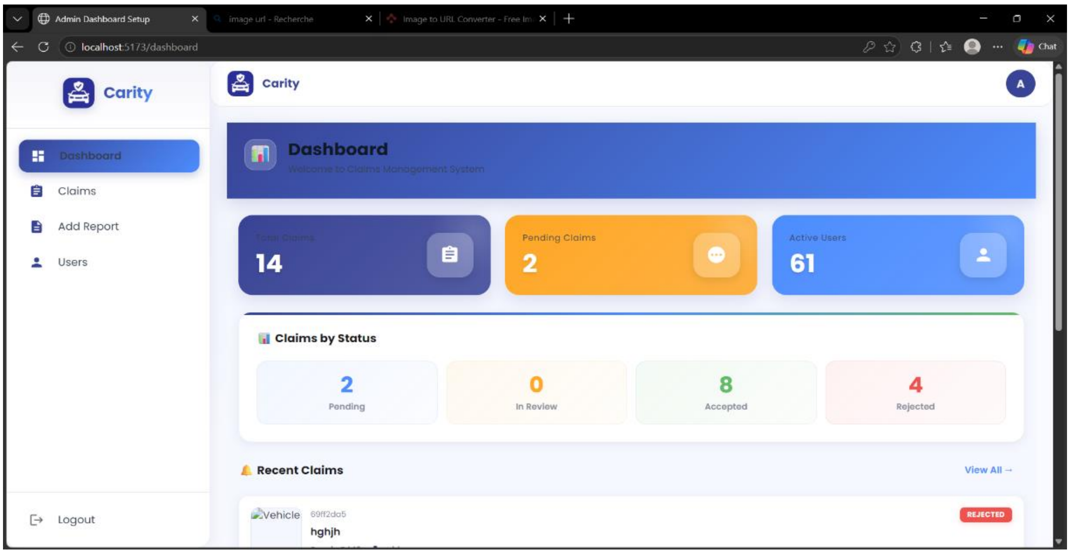


Figure 4.21 : Main Dashboard Claims Statistics Overview

The Users screen displays all registered customers in a card-based grid layout, where each card shows the user's full name, email address, phone number, account status, and total number of submitted claims.

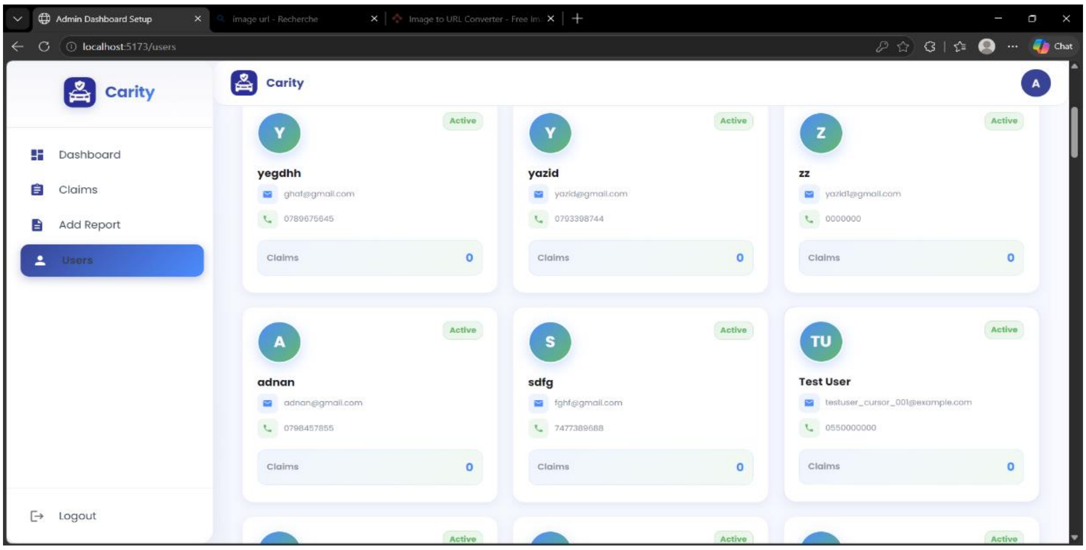


Figure 4.22 : Users Management Registered Customers List

4.7 Testing and Validation

4.7.1 Testing Approach

The system was validated through functional and integration testing methods applied across all three platforms. API endpoints were tested using Postman, mobile workflows were validated on physical Android devices, and the face verification microservice was tested independently using dedicated facial image pairs. Each test case was defined with a specific input, an expected result, and an observed outcome to ensure complete traceability of the validation process.

4.7.2 Functional Testing

Test	Feature	Input	Expected Result	Status
T01	Registration	Full name, email, phone, address	Profile saved, proceed to Step 2	Passed
T02	Registration	Vehicle type, model, usage, plate	Vehicle saved, proceed to Step 3	Passed
T03	Face Verification	Matching selfie	Verified: true, proceed to Step 4	Passed
T04	Face Verification	Non-matching image	Verified: false, access denied	Passed
T05	Policy Creation	Coverage dates confirmed	Policy saved, PDF generated	Passed
T06	User Login	Valid credentials	JWT token issued, home screen loaded	Passed
T07	User Login	Invalid credentials	Error message displayed	Passed
T08	Claim Submission	Accident data submitted	Claim stored, admin notified	Passed
T09	Claim Status Update	Admin updates to Approved	Customer notified in real time	Passed

Test	Feature	Input	Expected Result	Status
T10	Payment Confirmation	Successful webhook callback	Policy activated, transaction recorded	Passed
T11	PDF Generation	Approved policy	PDF document generated and downloadable	Passed
T12	Real-Time Notification	Claim status changed	Notification received instantly	Passed
T13	Logout	Click Logout	Session cleared, redirect to login	Passed

Table 4.11 : Functional Test Cases

Validation Summary

All thirteen defined test cases passed successfully across all tested platforms. The four-step onboarding flow, authentication, policy management, claims processing, payment integration, and face verification all operate correctly under normal conditions. Real-time notifications are delivered without measurable delay, and the administrative dashboard accurately reflects all system statistics and claim updates in real time.

4.8 Limitations

Although the proposed system fulfils its core objectives and all thirteen defined test cases passed successfully, several limitations were identified during development and testing:

- Face verification accuracy under poor lighting conditions. The DeepFace engine's matching accuracy degrades when facial images are captured in low-light environments or at extreme angles.
- Server-side file storage does not scale to high traffic environments. A cloud-based object storage solution such as AWS S3 would be more appropriate for production scale.
- Chargily Pay is limited to the Algerian market. The payment gateway supports only DZD denominated transactions, limiting the application's extensibility to other markets.

- The face verification channel is unencrypted in the development environment. A production deployment would require full TLS termination to secure biometric data during transmission.
- Single administrator account model. The current version supports only one administrator role without granular permission levels.
- Testing was conducted in a controlled local environment. System behavior under real world conditions such as high concurrent user load or very large claim volumes was not evaluated.

4.9 Future Work

From the limitations mentioned above, the following expansions are proposed for future development:

- Cloud Storage Integration: Moving uploaded files to a cloud storage service such as AWS S3 will improve the scalability and reliability of uploaded documents and personal photos.
- **AI-Powered Damage Assessment:** Integrating a computer vision model trained on vehicle damage datasets will allow the system to automatically estimate the severity of accident damage from uploaded photos.
- **Instant Notifications via Firebase Cloud Messaging:** Replacing Socket.io notifications with FCM will enable users to receive instant notifications even when the application is closed.
- **Expanded Role-Based Access Control:** Introducing multiple administrative roles, such as Claims Inspector and Financial Officer, will improve operational security within the control panel.
- **Offline Mode for the Mobile App:** Implementing a local caching layer using Hive or SQLite will allow customers to view insurance documents and claims history without an internet connection.
- **Full TLS/SSL Encryption for Microservices Connections:** Securing all communication channels between the mobile application and the micro-facial verification service will protect biometric data in production environments.
- **Multi-language support:** Adding Arabic and French will improve accessibility for Algerian users.

4.10 Conclusion

This chapter presented the complete implementation of the **Carity** vehicle insurance application, covering all phases from database design to functional testing and validation.

The five MongoDB collections Users, Policies, Claims, Transactions, and Notifications were detailed alongside their inter-collection relationships. The implementation covered the guided four-step onboarding flow, policy creation with automated PDF generation, GPS-enabled claims submission, payment processing via Chargily Pay, biometric face verification powered by DeepFace and Flask, real-time notifications via Socket.io, and a web-based administrative dashboard built with React.

All thirteen functional test cases passed successfully, confirming that the system behaves correctly under normal operating conditions. The identified limitations and proposed future extensions provide a clear roadmap for evolving the application toward a robust and production-ready insurance platform.

Conclusion

This work presented the design and implementation of Carity, a full-stack car insurance application developed to modernize and digitize the vehicle insurance process in Algeria. The proposed system addresses the key challenges of traditional insurance workflows by providing a seamless, secure, and fully digital experience for both customers and administrators.

Throughout this project, we covered the theoretical foundations of the main technologies adopted, analyzed the system requirements, designed the overall architecture, and implemented a complete multi-platform solution consisting of a Flutter mobile application, a React administrative dashboard, a Node.js backend API, and a Python Flask face verification microservice.

The system introduces a guided four-step onboarding flow that allows customers to register their profile, add their vehicle, verify their identity through biometric facial recognition, and select an appropriate insurance plan. Claims are submitted with GPS enabled accident location tracking and supporting photographic evidence, while administrators manage and review all submissions through a dedicated web-based dashboard. Payment processing is handled through the Chargily Pay gateway, supporting Algerian payment methods including EDAHABIA Card, CIB Card, and Chargily App. Real-time notifications powered by Socket.io keep customers informed of every update to their policies and claims.

The validation phase confirmed that all core functionalities operate correctly, with all defined functional test cases passing successfully across all tested platforms.

Despite the achievements of this work, several limitations were identified, including face verification sensitivity to poor lighting conditions, the absence of cloud-based file storage, and the restriction of the payment gateway to the Algerian market. These limitations outline a direction for future improvements, including the integration of cloud storage, AI powered damage assessment, push notifications via Firebase Cloud Messaging, role-based access control, and multi-language support for Arabic and French.

To conclude, Carity represents a step toward the digitalization of the insurance sector in Algeria, offering a solid architectural foundation that can be extended and scaled into a fully production ready platform.

References

[1] “Central Bank of the UAE, "Insurance Brochure," Available: https://www.centralbank.ae/media/jman22ug/insurance-brochure_a4_digital_a.pdf, Accessed: May 2026.,” p. 7.

[2] “A. Boukhlfa and Y. Lezgheb, "Conception et réalisation d'une application mobile pour la gestion des assurances des automobiles," Master Thesis, Abdelhafid Boussouf University of Mila, Computer Science Department, 2019.”.

[3] “M. Larabi, "Etude, Conception et Réalisation d’un Système Logiciel pour la Gestion des Assurances," Master Thesis, Université Abderrahmane Mira de Béjaïa, Faculté des Sciences Exactes, Département d’Informatique, 2017.”.

[4] “Société Nationale d’Assurance (SAA), "Assurance Automobile," Available: https://www.saa.dz, Accessed: 11-May-2026.”.

[5] “CAAT Assurance, "Guide des Garanties Automobiles," Online Document, Accessed: 11-May-2026.”.

[6] “AXA Assurance Algérie, "Les Garanties de l’Assurance Automobile," Online Document, Accessed: 11-May-2026.”.

[7] “Awras Aljazair, "تأمين السيارات في الجزائر: دليل شامل 2024,". Available: https://ar.awrasaljazair.com/تأمين-السيارات-في-الجزائر/”.

[8] “R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 8th ed. New York, NY, USA: McGraw-Hill Education, 2014.”.

[9] “B. Phillips, C. Stewart, K. Marsicano and B. Hardy, Android Programming: The Big Nerd Ranch Guide, 4th ed. Big Nerd Ranch Guides, 2019.”.

[10] “Google, "Android Developers Overview." Available: https://developer.android.com/about, Accessed: May 2026.”.

[11] “Apple Inc., "iOS - Apple Developer." Available: https://developer.apple.com/ios/, Accessed: May 2026.”.

[12] “A. Banafa, Mobile Applications and Digital Transformation. River Publishers, 2020.”.

[13] “C. Lee, H. K. Cheng and H. Cheng, "An empirical study of mobile commerce in the insurance industry: Task–technology fit and individual differences," Decision Support Systems, vol. 43, no. 1, pp. 95–110, 2007.”.

[14] “Société Nationale d’Assurance (SAA), "About SAA," Official Website. Available: https://www.saa.dz, Accessed: 2026.”.

[15] “G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language User Guide, 2nd ed. Addison-Wesley, 2005.”.

[16] “M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Addison-Wesley, 2003.”.

[17] “I. Sommerville, Software Engineering, 10th ed. Pearson Education, 2016.”.

[18] “Meta Open Source, "React Documentation." Available: https://react.dev/, Accessed: May 2026.”.

[19] “Microsoft, "TypeScript Documentation." Available: https://www.typescriptlang.org/docs/, Accessed: May 2026.”.

[20] “E. You, "Vite – Next Generation Frontend Tooling." Available: https://vitejs.dev/, Accessed: May 2026.”.

[21] “A. Wathan, "Tailwind CSS Documentation." Available: https://tailwindcss.com/docs, Accessed: May 2026.”.

[22] “MUI Team, "Material UI Documentation." Available: https://mui.com/material-ui/, Accessed: May 2026.”.

[23] “Axios Contributors, "Axios – Promise Based HTTP Client." Available: https://axios-http.com/docs/intro, Accessed: May 2026.”.

[24] “OpenJS Foundation, "Node.js Documentation." Available: https://nodejs.org/en/docs, Accessed: May 2026.”.

[25] “Express.js, "Express Documentation." Available: https://expressjs.com/, Accessed: May 2026.”.

[26] “MongoDB Inc., "MongoDB Documentation." Available: https://www.mongodb.com/docs/, Accessed: May 2026.”.

[27] “Mongoose Team, "Mongoose ODM Documentation." Available: https://mongoosejs.com/docs/, Accessed: May 2026.”.

[28] “Socket.IO Team, "Socket.IO Documentation." Available: https://socket.io/docs/v4/, Accessed: May 2026.”.

[29] “M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force (IETF), 2015.”.

[30] “N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," Proceedings of the USENIX Annual Technical Conference, 1999.”.

[31] “PDFKit Contributors, "PDFKit – A PDF Generation Library for Node.js." Available: https://pdfkit.org/, Accessed: May 2026.”.

[32] “Multer Contributors, "Multer – Node.js Middleware for Handling Multipart/Form-Data." Available: https://github.com/expressjs/multer, Accessed: May 2026.”.

[33] “Flutter Team, "Flutter Documentation." Available: https://docs.flutter.dev/, Accessed: May 2026.”.

[34] “Flutter Community, "camera | Flutter Package." Available: https://pub.dev/packages/camera, Accessed: May 2026.”.

[35] “Flutter Community, "image_picker | Flutter Package." Available: https://pub.dev/packages/image_picker, Accessed: May 2026.”.

[36] “Flask Documentation, "Flask Web Framework." Available: https://flask.palletsprojects.com/, Accessed: May 2026.”.

[37] “S. I. Serengil and A. Ozpinar, "DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework for Python," 2020.”.

[38] “Keras Team, "Keras Documentation." Available: https://keras.io/, Accessed: May 2026.”.

[39] “Gunicorn, "Gunicorn - Python WSGI HTTP Server." Available: https://gunicorn.org/, Accessed: May 2026.”.

[40] “The Pandas Development Team, "Pandas Documentation." Available: https://pandas.pydata.org/docs/, Accessed: May 2026.”.

[41] “NumPy Developers, "NumPy Documentation." Available: https://numpy.org/doc/, Accessed: May 2026.”.