الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and Popular Algerian Republic
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جـامـعـة الشهيد حمه لخضر الـوادي
Echahide Hamma Lakhdar University – El-Oued

Faculty of Exact Sciences كلية العلوم الدقيقة

Department of Computer Science قسم الإعلام الآلي

Thesis

Presented for the attainment of a license
Degree in computer science

Presented By: Bekkari Haithem
Drihem Abdelmoumen

Theme

A mobile app for police to track car status

Presented on ... - ... - 2025

Professor :……	MCA	President
Professor :……	MAA	Decided
Professor :	Mohammed Amine Yagoub	Supervisor

Academic Year: 2024/2025

شكر وتقدير

أود أن أعبر عن خالص شكري وتقديري لإدارة الكلية وأعضاء هيئة التدريس الذين قدموا لي المعرفة والإرشاد و البيئة الأكاديمية المناسبة لإعداد هذا البحث. كما أخص بالشكر مشرفي الأكاديمي الدكتور محمد أمين يعقوب على دعمه المستمر وإرشاداته القيمة.

لا يسعني إلا أن أشكر عائلتي العزيزة التي كانت مصدر دعمي وإلهامي خلال هذه الرحلة، وأصدقائي الذين وقفوا بجانبي دائمًا. كما أشكر كل من ساهم في إنجاح هذا البحث، سواء بالنصيحة أو المساندة.

Summary

The objective of our final year project, presented in this report, is the design and development of a mobile application dedicated to the police for tracking and managing the status of vehicles. This project aims to address several issues, such as the quick verification of vehicle information, real-time detection of infractions, and the optimization of law enforcement interventions.

The application will enable police officers to instantly access registered vehicle information, report anomalies, and facilitate the verification of administrative documents. With an intuitive and user-friendly interface, the application will simplify data management and enhance the efficiency of police services.

For the design, we used the UML modeling language, while the database management is handled by a MySQL server.

Keywords: Mobile Application, Police, Vehicle Management, Services, Car Status Tracker, SQL, MySql, Kotlin, Jetpack Compose, Adobe Figma, Node.js, Express, Render(for api deployment), Clever Cloud, Api.

Contents

List of Figures

GENERAL INTRODUCTION

Today, the world is witnessing significant technological advancements in all sectors, particularly in road safety and vehicle tracking. Modern technologies now enable law enforcement agencies to use mobile applications to monitor and manage vehicle status in real time.

Police officers often face numerous challenges, such as quickly identifying vehicles, verifying documents, and detecting infractions. Thanks to computing and web services, it is now possible to enhance the efficiency of these operations by automating the process of vehicle control and tracking.

In this context, we have been led to design, develop, and implement a mobile application dedicated to vehicle management by the police. This application aims to centralize information, provide instant access to vehicle data, and facilitate officers' interventions in the field.

Our application is designed to improve vehicle management and optimize law enforcement operations to enhance the reliability of inspections, reduce response times, and minimize human errors.

This project consists of three chapters presented as follows:

- The first chapter is dedicated to the theoretical presentation and functional requirements.
- Chapter II provides several UML diagrams and specifications.
- Chapter III allows for the visualization and understanding of different system interfaces.

Finally, we will conclude this report with a summary of the work accomplished and an introduction to future perspectives for this project.

SECTION I
SYSTEM CONTEXT

1. Introduction

This chapter provides a general overview of the work context and the objectives of our final year project. We will begin by outlining some of the challenges faced by law enforcement and the importance of an information system for police operations.

2. The Importance of Police in Taking Security Measures to Reduce Violations of Vehicle Condition Laws

The police play a crucial role in ensuring road and vehicle safety by implementing security measures to reduce violations related to vehicle conditions. Ensuring that vehicles are roadworthy is essential to prevent accidents caused by mechanical failures or the deterioration of critical components.

The police conduct regular inspection campaigns to detect vehicles that do not meet legal standards, such as those with faulty brakes, lighting, or tires, which may pose risks to drivers and pedestrians. They also impose fines and penalties on violators to ensure compliance with laws designed to enhance road safety.

In addition, the police contribute to raising awareness among drivers about the importance of regular vehicle maintenance through educational and awareness programs. They also utilize modern technologies, such as smart cameras and electronic inspection devices, to monitor vehicle conditions and effectively detect violations.

Thus, the role of the police in enforcing security measures is not limited to imposing laws but also includes promoting a culture of road safety awareness, which helps reduce accident rates and ensures a safer traffic environment for everyone.

3. Role of PolicePlus System for Law Enforcement

A digital system can significantly improve police operations by offering:

➢Automatic license plate recognition and vehicle status tracking.

➢Quick and efficient vehicle information retrieval.

➢Secure storage of police records with encryption.

➢Integration with external databases (e.g., insurance, stolen vehicles, tickets).

➢Real-time data updates and synchronization.

➢Enhanced communication between police departments and external entities.

4. How the Police Monitor Vehicle Condition and Detect Violations

Monitoring the condition of vehicles is a key responsibility of the police to ensure road safety and reduce accidents. Police officers rely on a set of documents and standards to determine whether a vehicle complies with traffic laws. This document highlights five main factors that help the police monitor vehicle condition and detect potential violations.

Files needed to monitor vehicle condition:

1.1 Driver's License

The driver's license is an essential document that drivers must carry while driving. It includes important details such as the license number, nickname, and name of the driver. The police verify the validity of the license to ensure that the driver is legally qualified to operate a vehicle. If the license is expired or invalid, it is considered a violation.

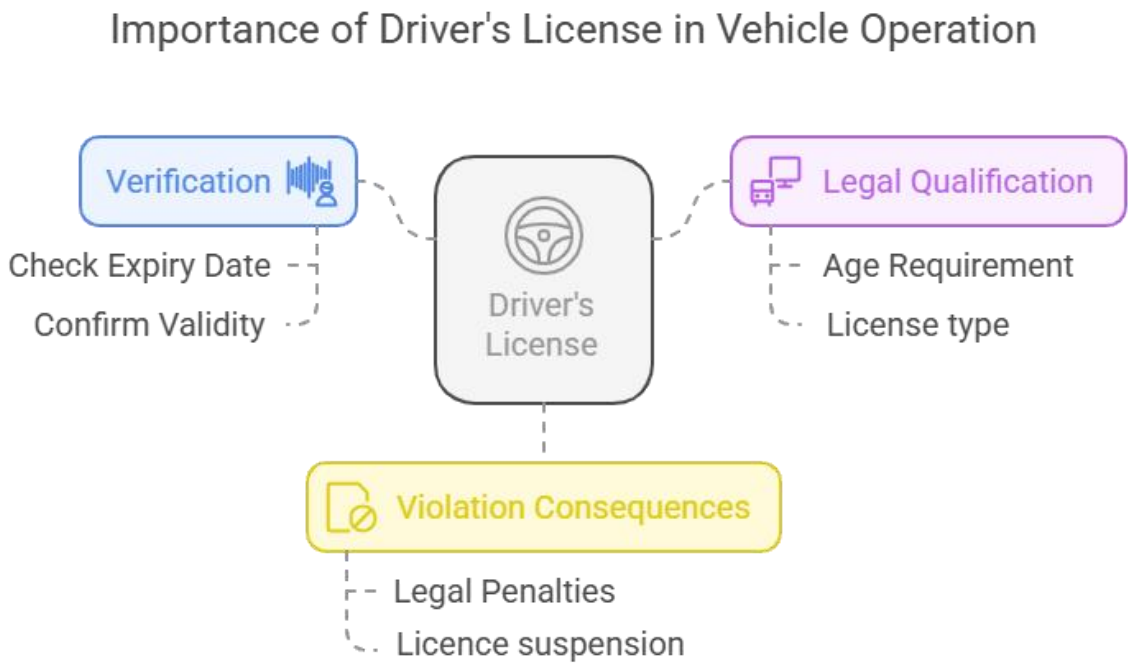


Figure I.1: Driver License

4.2. Vehicle Registration Certificate

The vehicle registration certificate (also known as the gray card) proves ownership of the vehicle and contains important information such as the matriculation. The police check this document to confirm that the vehicle is legally registered and not stolen. Any discrepancies or tampering with the information can result in fines.

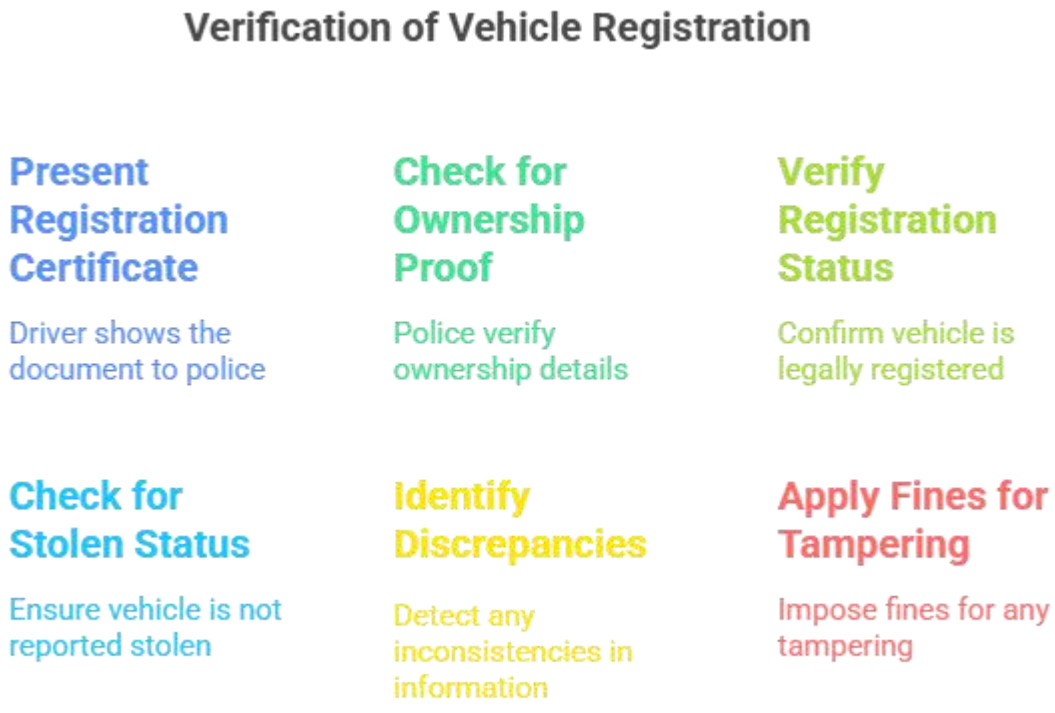


Figure I.2: Registration Certificate

4.3 Insurance Certificate

A valid insurance certificate is a legal requirement for drivers. It includes important details such as the validity period, specified as "Valid from Date to Date." The police verify the presence of an active insurance policy, as driving without insurance is a legal violation. Insurance protects both drivers and pedestrians in the event of an accident.

Figure I.3: Insurance Certificate

4.4 Periodic Technical Inspection

Laws in many countries require vehicles to undergo regular technical inspections to ensure their safety and efficiency. The police check for a valid technical inspection certificate, which includes details such as the Subsequent Monitoring Date, to confirm that the vehicle meets the required mechanical and safety standards. Failing to undergo periodic inspections can result in penalties.

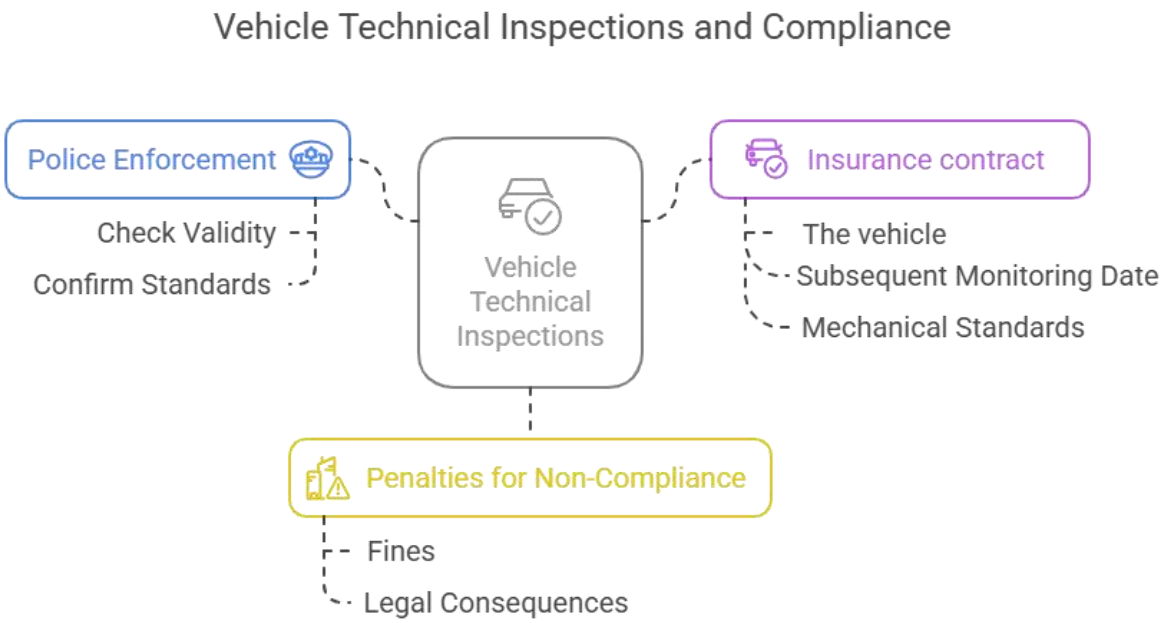


Figure I.4: Technical Inspection

4.5 Tax Receipt

The tax receipt serves as proof that the necessary vehicle taxes have been paid. The police verify this document to ensure that the driver complies with financial regulations related to vehicle ownership. Failure to pay taxes can lead to fines or even vehicle impoundment.

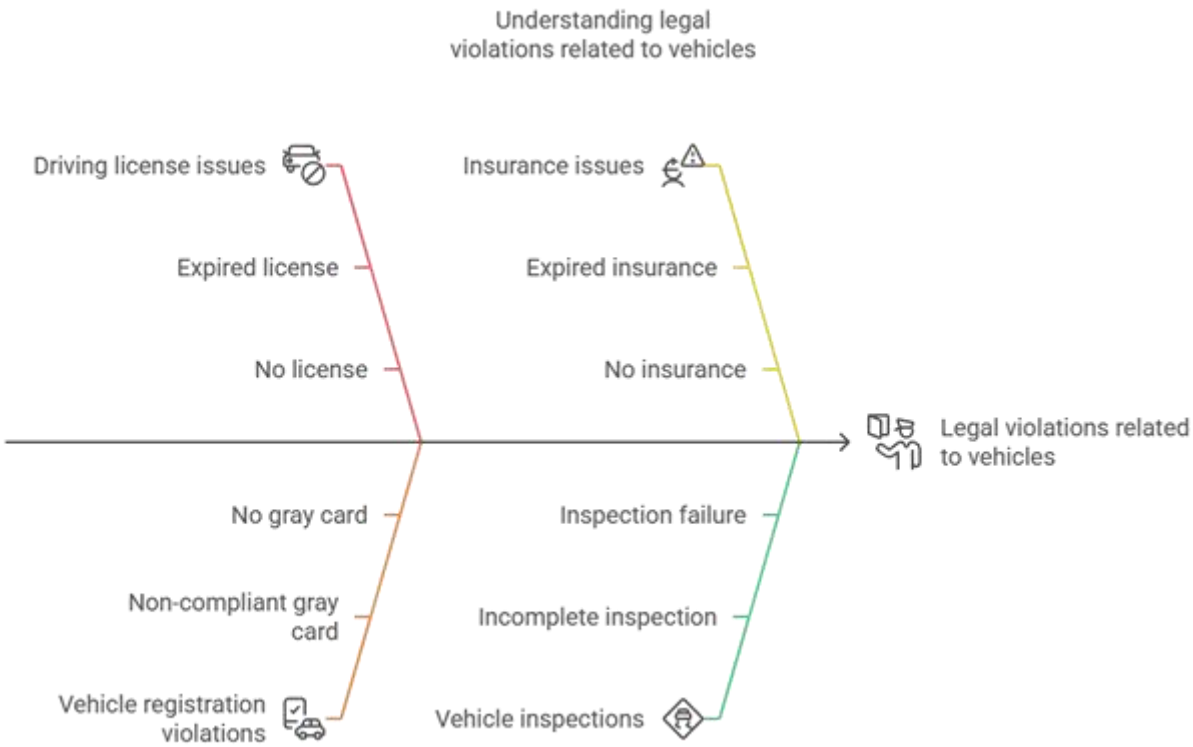


Figure I.5: Diagram of different possible drivers violations

The police rely on these five key documents and standards to maintain road safety and ensure that all drivers comply with traffic laws. It is essential for drivers to keep their documents valid and up to date to avoid violations and penalties.

5. Challenges in Police Operations

Law enforcement agencies often encounter various difficulties when managing vehicle-related data, including:

➢Difficulty in searching for vehicle records manually.

➢Time-consuming processes leading to delays in operations.

➢Risk of document loss or damage in paper-based systems.

➢Limited storage capacity for physical records.

➢Inaccurate or outdated record-keeping.

➢Challenges in statistical analysis and reporting.

➢Difficulty in accessing real-time vehicle information.

➢Lack of an efficient system for verifying license plates and retrieving related data.

Vehicle inspections are still largely conducted manually, resulting in reduced efficiency.

Is there a smart solution that simplifies and improves the inspection process for law enforcement officers?

6. Conclusion

In this first chapter, we provided an overview of the challenges associated with manual vehicle inspection. We then focused on identifying key issues that hinder efficiency and accuracy.

In the next chapter, we will analyze the requirements and explore how our proposed solution can address these challenges.

SECTION II
SYSTEM DESIGN

1. Introduction

In this chapter, we present the design of our system. The modeling is done using UML (Unified Modeling Language).

The main diagrams used are:

- Use Case Diagrams
- Sequence Diagrams
- Class Diagrams

---

2. Needs Analysis

2.1 UML Definition:

UML (Unified Modeling Language) is a standard object-oriented modeling language used to specify, visualize, construct, and document the components and structure of an information system.

2.2 UML Modeling for the PolicePlus Mobile App:

In this section, we model the most important features of the **PolicePlus mobile application**, which is designed to help police officers track vehicle status using license plate recognition.

We use UML diagrams to:

- Illustrate user interactions with the system
- Describe internal system behavior
- Visualize data flow and database relations

2.3 Identification of Actors:

Actor	Role
Police Officer (App User)	- Register and log in to the app - Scan license plates - View car status - File a ticket - Report a stolen car - Resume ticket drafts
Traffic management	- Populate police database with car and officer records - Search and consult car data and violations
Car Owner	- The system displays data about car owners

2.4 Use Case Diagrams:

2.4.1 UML Diagrams for Police Officer

A - Use Case Diagram

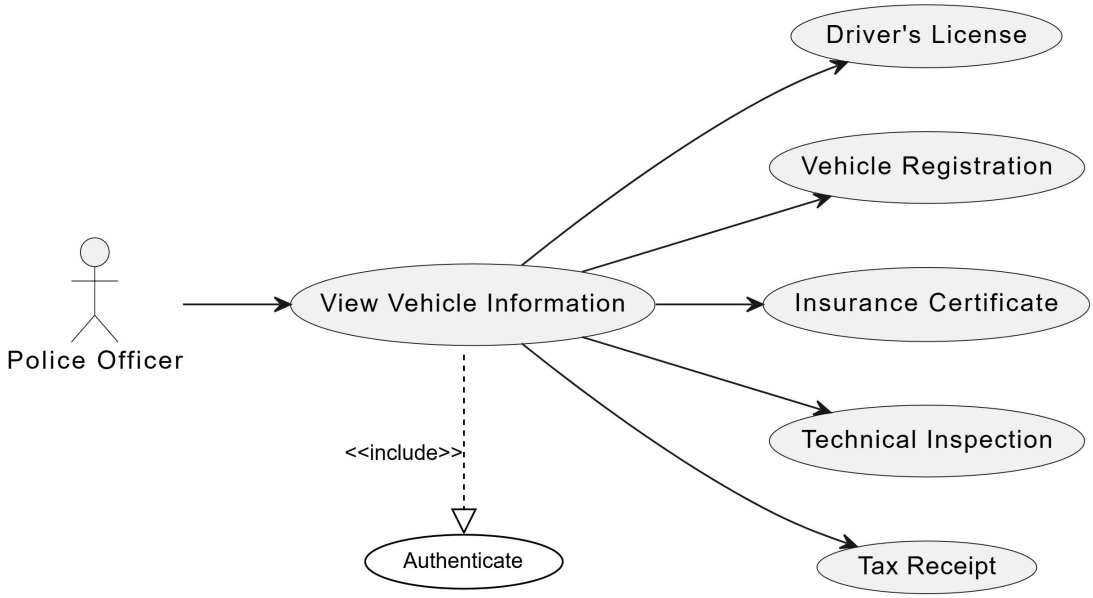


Figure II.1: Use Case Diagram "Police Officer"

This use case diagram shows how a Police Officer interacts with the system to authenticate and then view vehicle information. After authentication, the officer can access details such as the Driver's License, Vehicle Registration, Insurance Certificate, Technical Inspection, and Tax Receipt to verify a vehicle's legal status.

B - Sequence Diagram

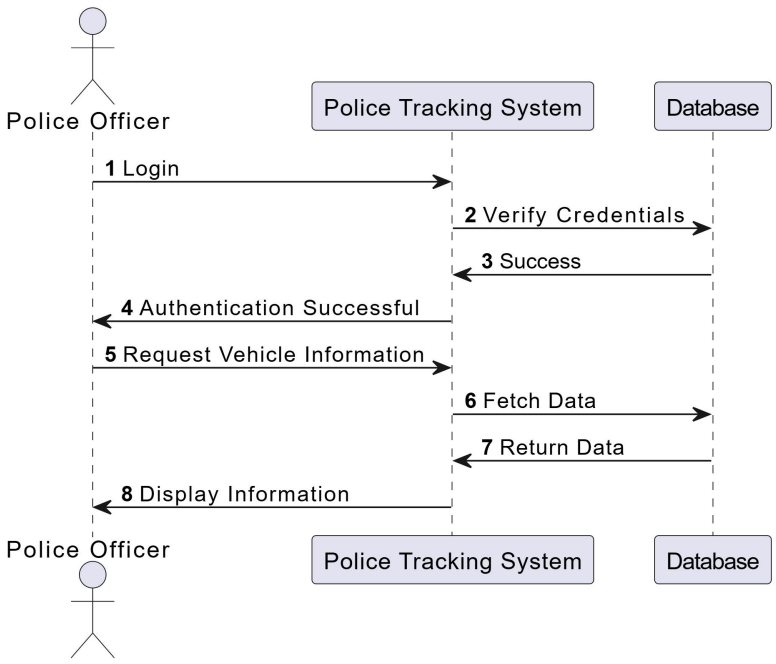


Figure II.2: Sequence Diagram "Police Officer"

C - Class Diagram

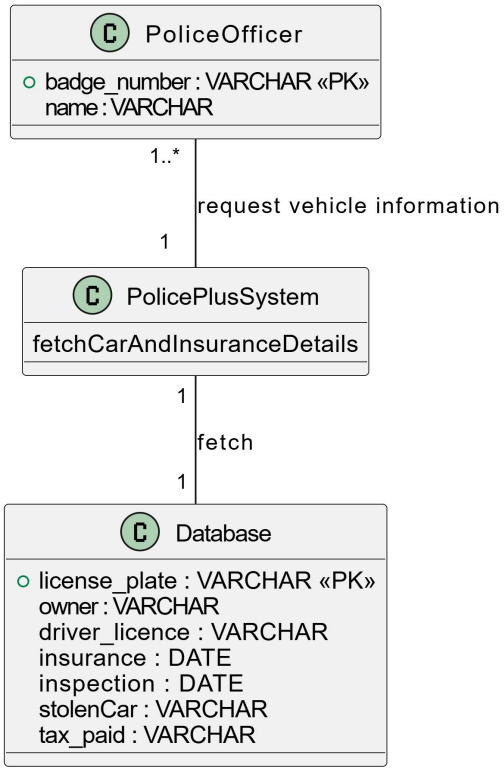


Figure II.3: Class Diagram "Police Officer"

2.4.2 UML Diagrams for Car status management

A - Use Case Diagram

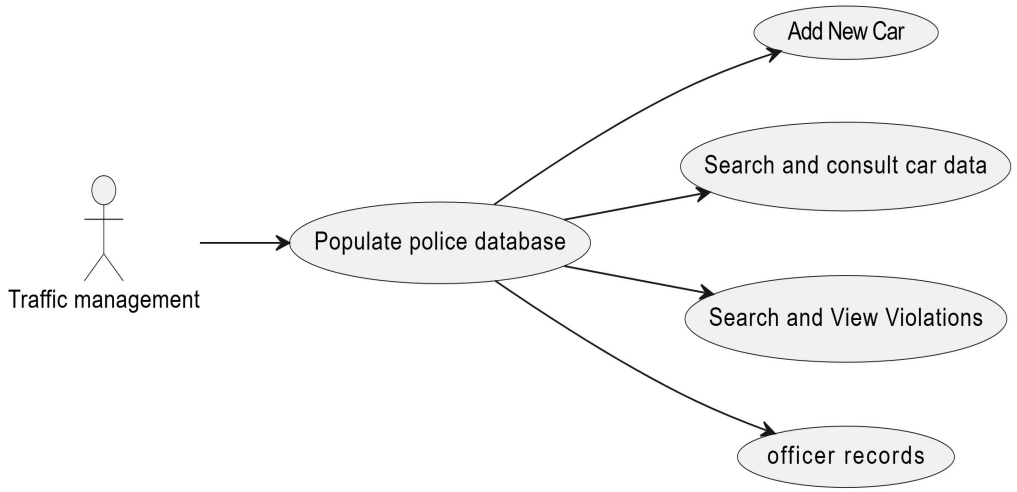


Figure II.4: Use Case Diagram "Car status"

This use case diagram illustrates how the Traffic Management actor interacts with the system to populate the police database. Through this main use case, traffic management can perform tasks such as adding new cars, searching and consulting car data, viewing violations, and managing officer records, ensuring the database remains accurate and up to date.

B - Sequence Diagram

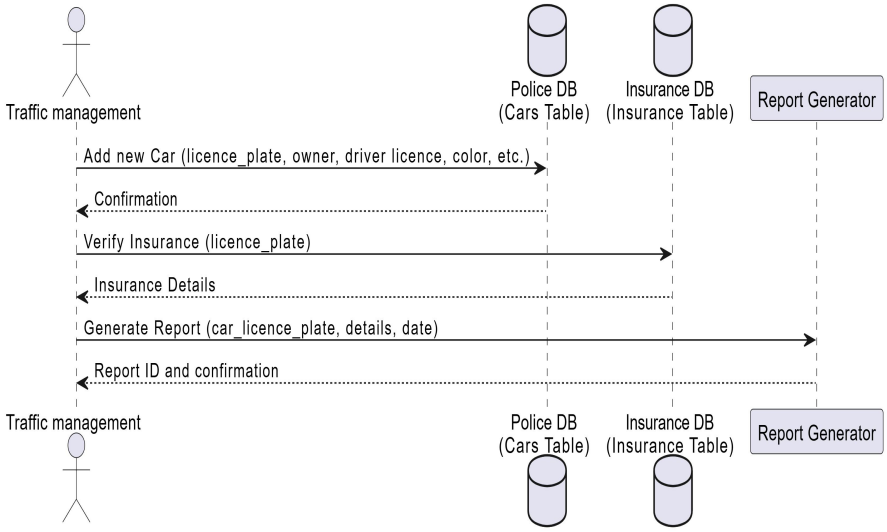


Figure II.5: Sequence Diagram "Car status"

C - Class Diagram

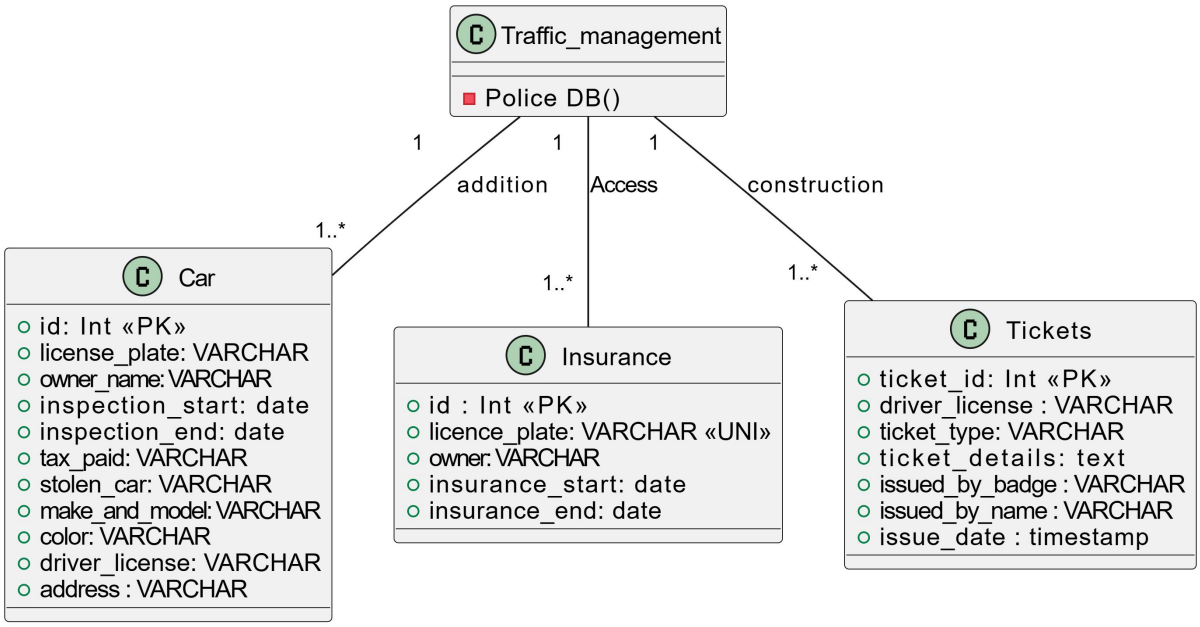


Figure II.6: Class Diagram "Car status"

2.4.3 UML Diagrams for Accounts management

A - Use Case Diagram

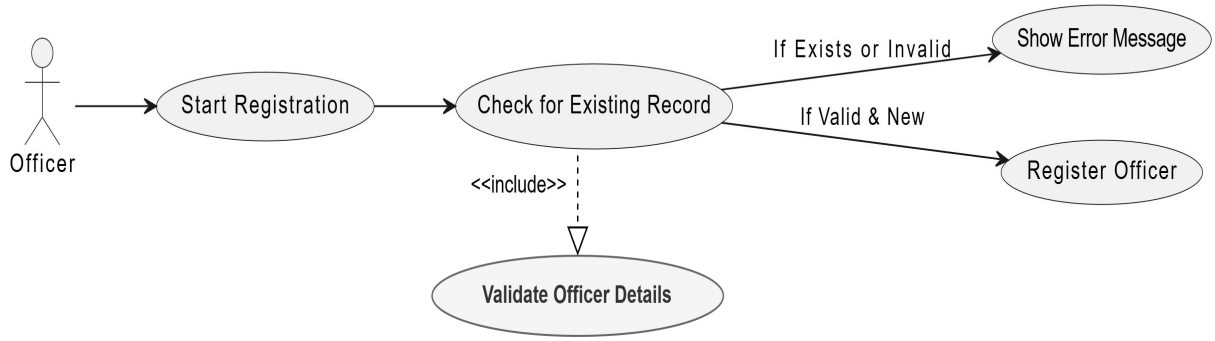


Figure II.7: Use Case Diagram "Registration"

This use case diagram represents the Officer Registration Process. The Officer starts the registration, which triggers a Check for Existing Record process. This process includes Validating Officer Details to ensure correctness. If the record already exists or is invalid, an Error Message is shown. If the record is valid and new, the system proceeds to Register the Officer successfully.

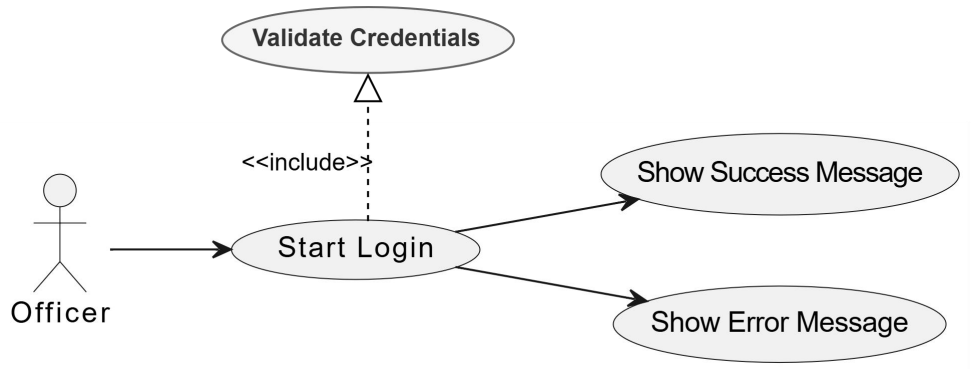


Figure II.8: Use Case Diagram "Login"

This use case diagram represents the Officer Login Process. The Officer initiates the Start Login process, which includes Validating Credentials to ensure authentication. If the credentials are correct, a Success Message is displayed. If they are incorrect, an Error Message is shown.

B - Sequence Diagram

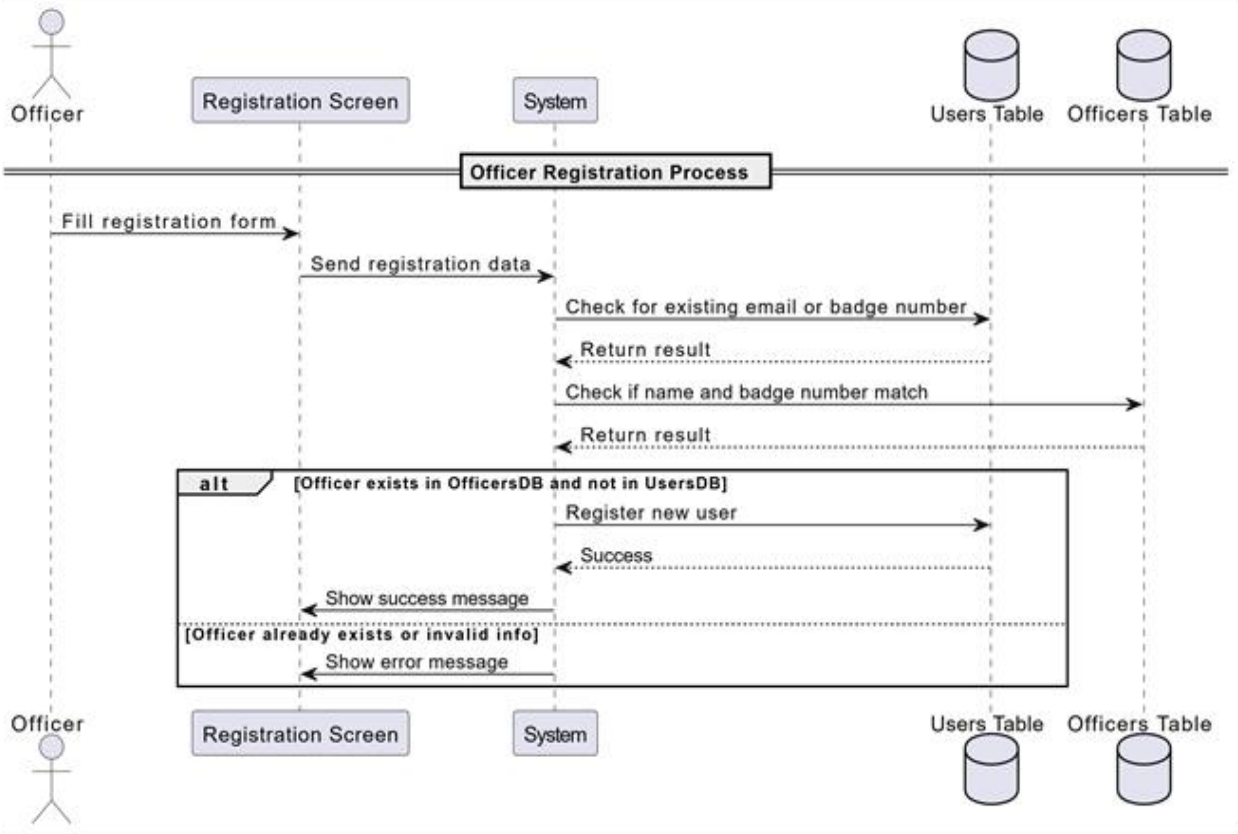


Figure II.9: Sequence Diagram "Accounts management"

C - Class Diagram

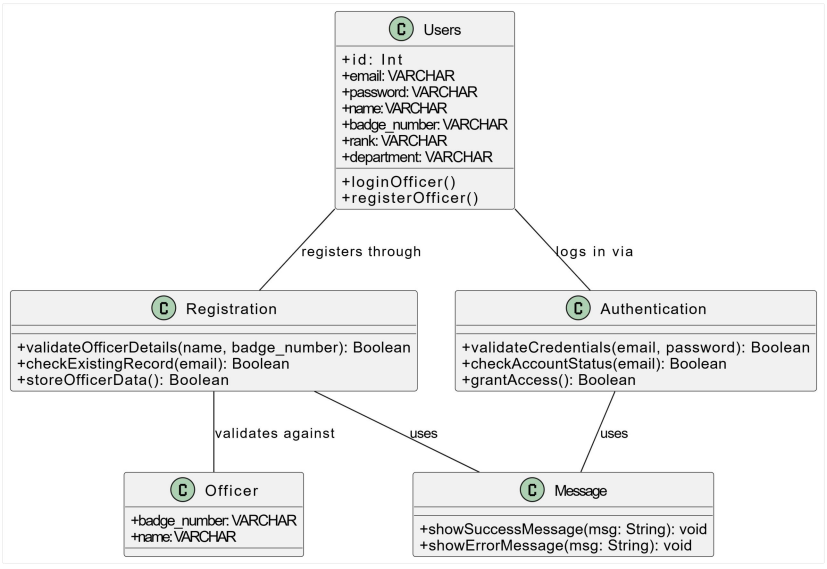


Figure II.10: Class Diagram "Accounts management"

2.4.4 UML Diagrams for Declaration of an accident

A - Use Case Diagram

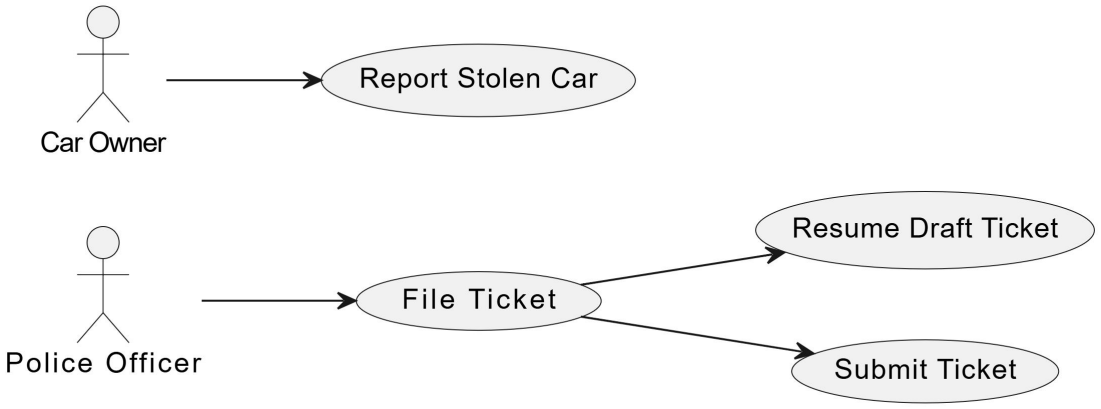
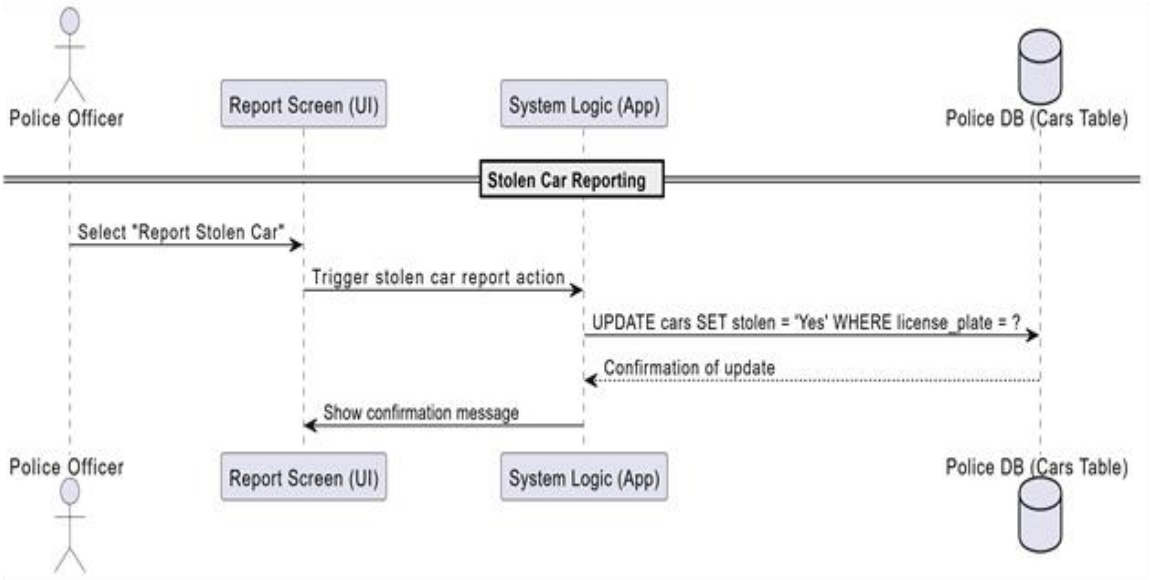


Figure II.11: Use Case Diagram "Declaration of an accident"

The diagram shows that police officers can file a ticket, resume a draft, or submit it. Car owners can report a stolen vehicle.

B - Sequence Diagram

-Stolen Car



-Ticket Filing

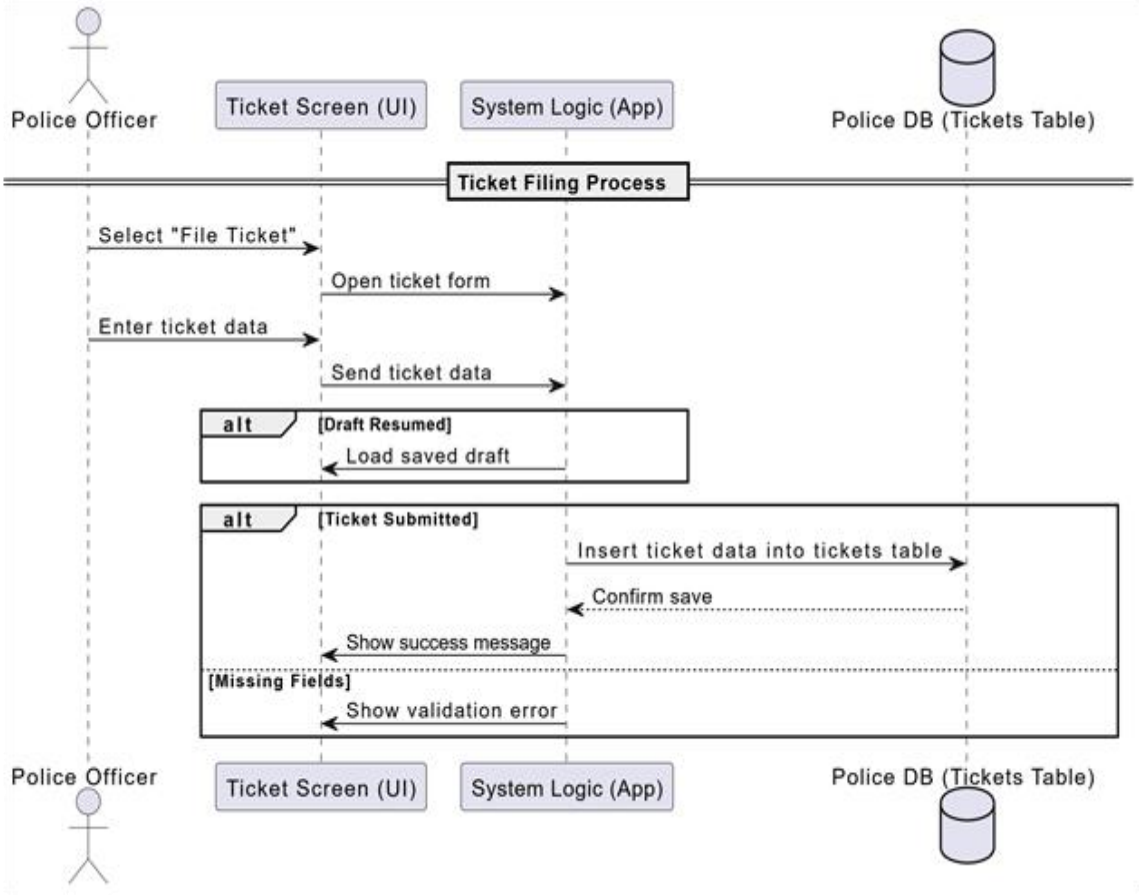


Figure II.12: Sequence Diagram "Declaration of an accident"

C - Class Diagram

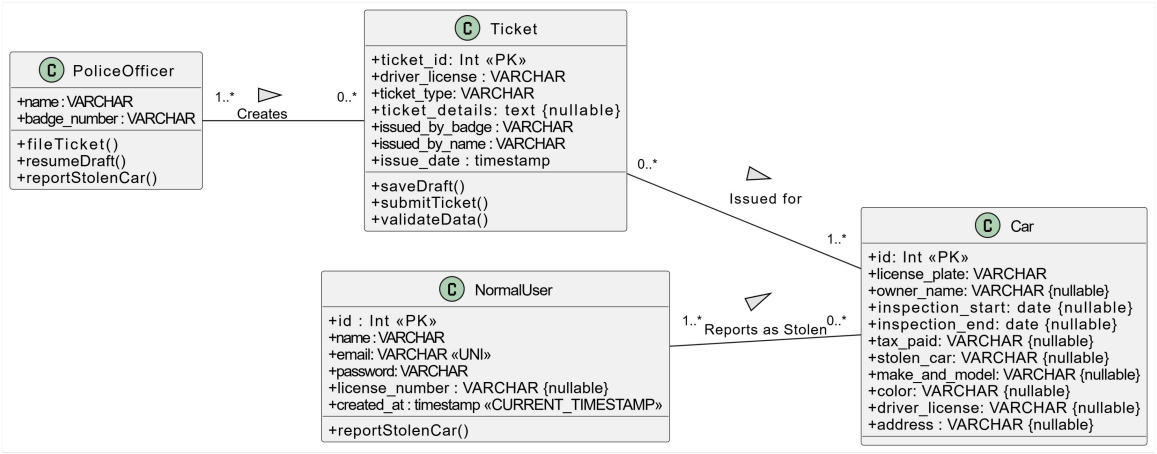


Figure II.13: Class Diagram "Declaration of an accident"

📌2.4.5 Relationships

Police Database:

• users: Contains information about officer users of the application such as email, password, name, rank, department, and badge number.
• officers: Includes names and badge numbers of real officers, used for identity verification during registration.
• normal_users: Represents regular users (e.g., drivers), including name, email, password, and driver's license number.
• cars: Includes vehicle data such as make and model, color, license plate, owner name, driver license, inspection start/end dates, tax payment status, and stolen status.
• tickets: Represents traffic tickets issued by officers to drivers.

Insurance Database:

• Insurance: Contains insurance information related to vehicles, including license plate, owner name, insurance start and end dates.

Relationships:

• Each vehicle in the Police Database is matched with the Insurance Database using the **license_plate**.
• A user who issues tickets must be a verified officer by matching their **name and badge number** with the officers table.
• A user of type "officer" can issue multiple tickets (One-to-Many: users → tickets).
• Each ticket is linked to a regular user (driver) through the **driver's license** (Many-to-One: tickets → normal_users).
• Vehicles can be associated with regular users through the **driver license** field.
• A regular user can report a stolen vehicle by providing their **driver's**

license and the stolen status (Yes/No), which updates the stolen_car field in the cars table.

• Each vehicle can have one associated insurance record (One-to-One: cars → Insurance).

2.5 Class Diagram :

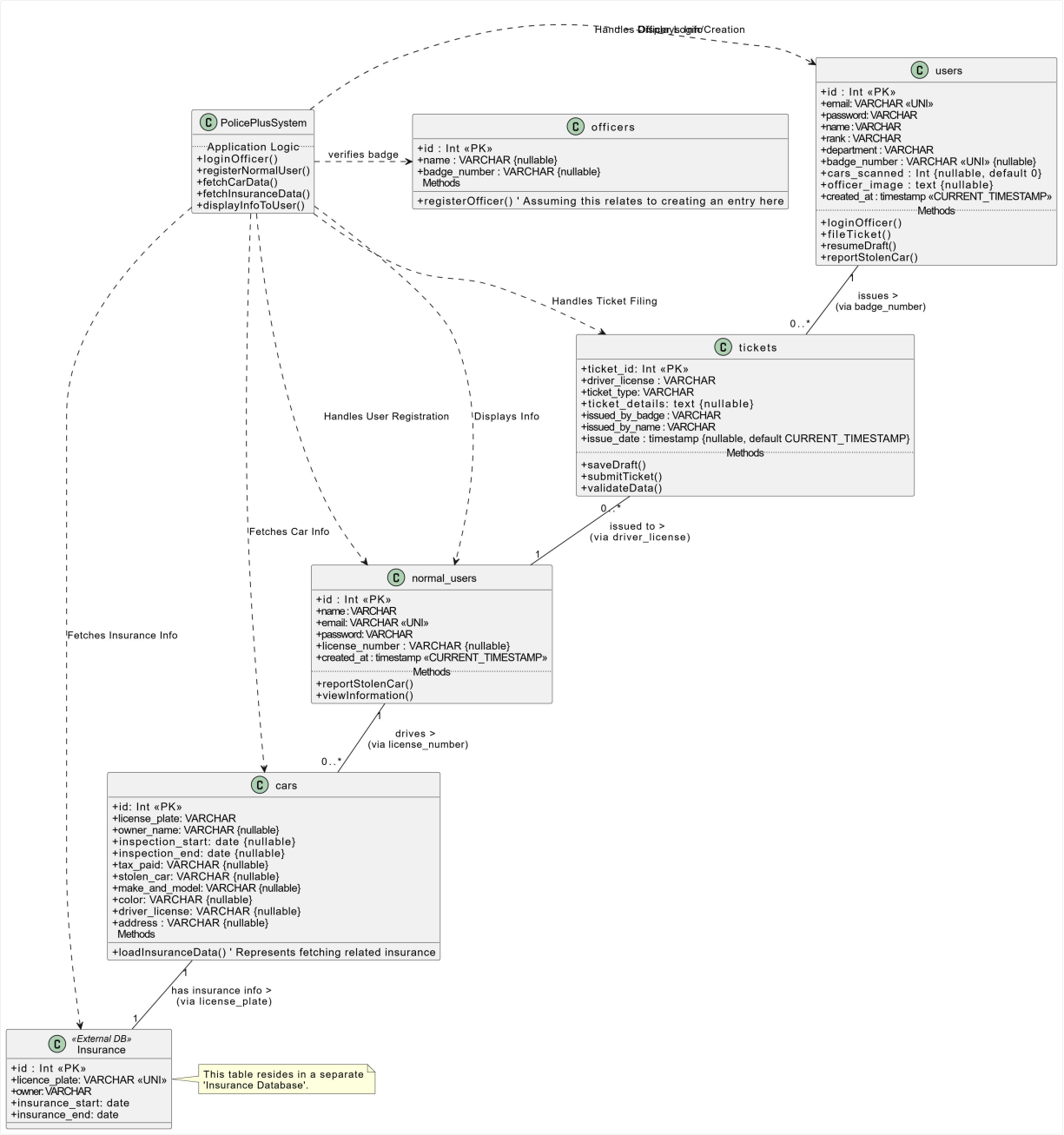


Figure II.14: Class Diagram

2.6 Moving from class schema to databases :

users(id, email, password, name, rank, department, badge_number, cars_scanned, officer_image, created_at)

tickets(ticket_id, driver_license, ticket_type, ticket_details, issued_by_badge, issued_by_name, issue_date,# driver_license, #issued_by_badge)

officers(id, name, badge_number)

normal_users(id, name, email, password, license_number, created_at)

cars(id, license_plate, owner_name, inspection_start, inspection_end, tax_paid, stolen_car, make_and_model, color, driver_license, address, #license_number)

Insurance(id, licence_plate, owner, insurance_start, insurance_end, #licence_plate)

2.7 Data dictionary :

Category	Element	Coding	Data type	Value
User	ID	id	Int	PK
	Email	email	VARCHAR	255
	Password	password	VARCHAR	255
	Name	name	VARCHAR	100
	Rank	rank	VARCHAR	50
	Department	department	VARCHAR	100
	Badge Number	badge_number	VARCHAR	50 (nullable, UNI)
	Cars Scanned	cars_scanned	Int	default 0, nullable
	Officer Image	officer_image	TEXT	Nullable

	Created At	created_at	timestamp	CURRENT_TIMESTAMP
Ticket	ID	ticket_id	Int	PK
	Driver License	driver_license	VARCHAR	50 (FK to normal_users)
	Ticket Type	ticket_type	VARCHAR	50
	Ticket Details	ticket_details	TEXT	Nullable
	Badge Number	issued_by_badge	VARCHAR	50 (FK to users)
	Officer Name	issued_by_name	VARCHAR	100
	Issue Date	issue_date	timestamp	default CURRENT_TIMESTAMP, nullable
Officer	ID	id	Int	PK
	Name	name	VARCHAR	100, nullable
	Badge Number	badge_number	VARCHAR	50, nullable
User (normal users)	ID	id	Int	PK
	Name	name	VARCHAR	100
	Email	email	VARCHAR	255 (UNI)
	Password	password	VARCHAR	255
	License Number	license_number	VARCHAR	50, nullable
	Created At	created_at	timestamp	CURRENT_TIMESTAMP
Car	ID	id	Int	PK
	License Plate	license_plate	VARCHAR	20
	Owner Name	owner_name	VARCHAR	100, nullable
	Inspection Start	inspection_start	DATE	Nullable
	Inspection End	inspection_end	DATE	Nullable
	Tax Paid	tax_paid	VARCHAR	10, nullable
	Stolen Car	stolen_car	VARCHAR	10, nullable
	Make and Model	make_and_model	VARCHAR	100, nullable
	Color	color	VARCHAR	50, nullable
	Driver License	driver_license	VARCHAR	50, nullable

	Address	address	VARCHAR	255, nullable
Insurance	ID	id	Int	PK
	License Plate	licence_plate	VARCHAR	20 (UNI, FK to cars)
	Owner	owner	VARCHAR	100
	Insurance Start	insurance_start	DATE	
	Insurance End	insurance_end	DATE	

3. Conclusion

This chapter allowed us to present the approach followed to address the problem in question, whether through the description of the system hierarchy or the diagrams outlining the structure of the app. However, the practical implementation still remains — which will be the focus of the next chapter.

SECTION III
SYSTEM IMPLEMENTATION

1. Introduction

In this chapter, we will discuss the tools and specific programming languages used in the development and implementation of the website. We will also present the graphical interfaces of this application.

2. Tools Used

2.1 Room database :

Room is a persistence library introduced as part of the Android Jetpack suite to simplify working with SQLite databases. It provides:

- A structured approach to manage data.
- Compile-time checks for SQL queries.
- Seamless integration with modern Android components like LiveData and Coroutines. [1]

2.2 Android Studio :

Android Studio is the official Integrated Development Environment (IDE) for Android app development. Based on the powerful code editor and developer tools from IntelliJ IDEA , Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Live Edit to update composables in emulators and physical devices in real time
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks. [2]

3. Programming languages and technologies used

3.1 Kotlin Language :

Kotlin is an open-source, statically-typed programming language that supports both object-oriented and functional programming. Kotlin provides similar syntax and concepts from other languages, including C#, Java, and Scala, among many others. Kotlin does not aim to be unique, instead, it draws inspiration from decades of language development. It exists in variants that target the JVM (Kotlin/JVM), JavaScript (Kotlin/JS), and native code (Kotlin/Native). [3]

3.2 Mysql :

"SQL", the acronym for Structured Query Language. A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language that programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups. [4]

3.3 Jetpack Compose :

Jetpack Compose is Android’s recommended modern toolkit for building native UI. It simplifies and accelerates UI development on Android. Quickly bring your app to life with less code, powerful tools, and intuitive Kotlin APIs. [5]

3.4 Dagger Hilt :

Dagger Hilt is a dependency injection library for Android, built on top of Dagger 2. Hilt is designed to simplify the implementation of dependency injection in Android apps by reducing boilerplate code and providing seamless integration with Android components

such as activities, fragments, services, and ViewModels. It leverages the power of Dagger under the hood while offering a more concise and intuitive API for developers. [6]

3.5 Retrofit :

Retrofit is a type-safe HTTP client for Android and Java, library that simplifies this process by providing a clean and efficient way to make API calls. [7]

3.6 Nodejs :

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project! Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant. [8]

3.7 Express :

Express is an open-source web application framework for Node.js. It provides a robust set of features for building web and mobile applications, including routing, middleware, template engines, seamless database integration, and a wealth of features for developing advanced features and functions. [9]

3.8 Render :

Render is a cloud-based platform that simplifies the process of deploying and hosting web applicatoins, APIs, static sites, and more. It automates tasks like building, deploying, and scaling applications, making it easier for us to bring our project to production.[10]

3.9 Clever Cloud :

Clever Cloud provides an automated hosting platform for developers. Deploy your app easily and launch dependencies without having to worry about the infrastructure set up. Follow this guide to get ready to deploy quickly as you learn the basics of Clever Cloud. [11]

3.10 Postman :

Postman is an all-in-one API platform for building and working with APIs. It takes the pain out of every stage of the API lifecycle—from designing and testing to delivery and monitoring. Built for

teams, Postman makes it easy to collaborate, stay organized, and build secure, reliable APIs faster. [12]

3.11 Figma :

Figma is a collaborative web application for interface design, with additional offline features enabled by desktop applications for macOS and Windows. The feature set of Figma focuses on user interface and user experience design. [13]

3.12 Git & GitHub :

Git is a version control system that intelligently tracks changes in files. Git is particularly useful when you and a group of people are all making changes to the same files at the same time.
GitHub is a cloud-based platform where you can store, share, and work together with others to write code. [14]

4. Summary table

Component	Technology Used
Mobile App	Kotlin, Jetpack Compose
Backend APIs	JavaScript (Node.js + Express)
Database	MySQL
UI/UX Design	Adobe Figma
Deployment	Render (APIs), Clever Cloud (Databases)
Local Data Storage	Room (Android SQLite ORM)
Dependency Injection	Dagger Hilt
Network Communication	Retrofit, Axios
Security	Bcrypt (password hashing), JWT (token authentication)
IDEs	Android Studio, Visual Studio Code
Version Control	Git, GitHub
API Testing	Postman

5. Presentation of the Developed Application

Before starting the development, it was necessary to prepare the essential tools. After familiarizing ourselves with the development environment, we first integrated the necessary libraries and then defined the usage scenario of the application. The different stages are described below:

5.1 Scenario:

Our application is designed for police officers to facilitate the management and tracking of vehicles. It allows officers to quickly access vehicle information by scanning or entering a license plate number.

The application is connected to a centralized database where vehicle information is stored, including the registration certificate, the owner's driver's license, insurance details, technical inspection status, and road tax payment.

With this application, law enforcement can verify the administrative status of vehicles in real time, detect potential violations, and optimize their field interventions. The application requires an internet connection to query the remote database and ensure real-time updates of information.

Figure III.1: Application

5.2 Description of the Application Interface:

In the following, we present the different interfaces of the application, detailing each screenshot.

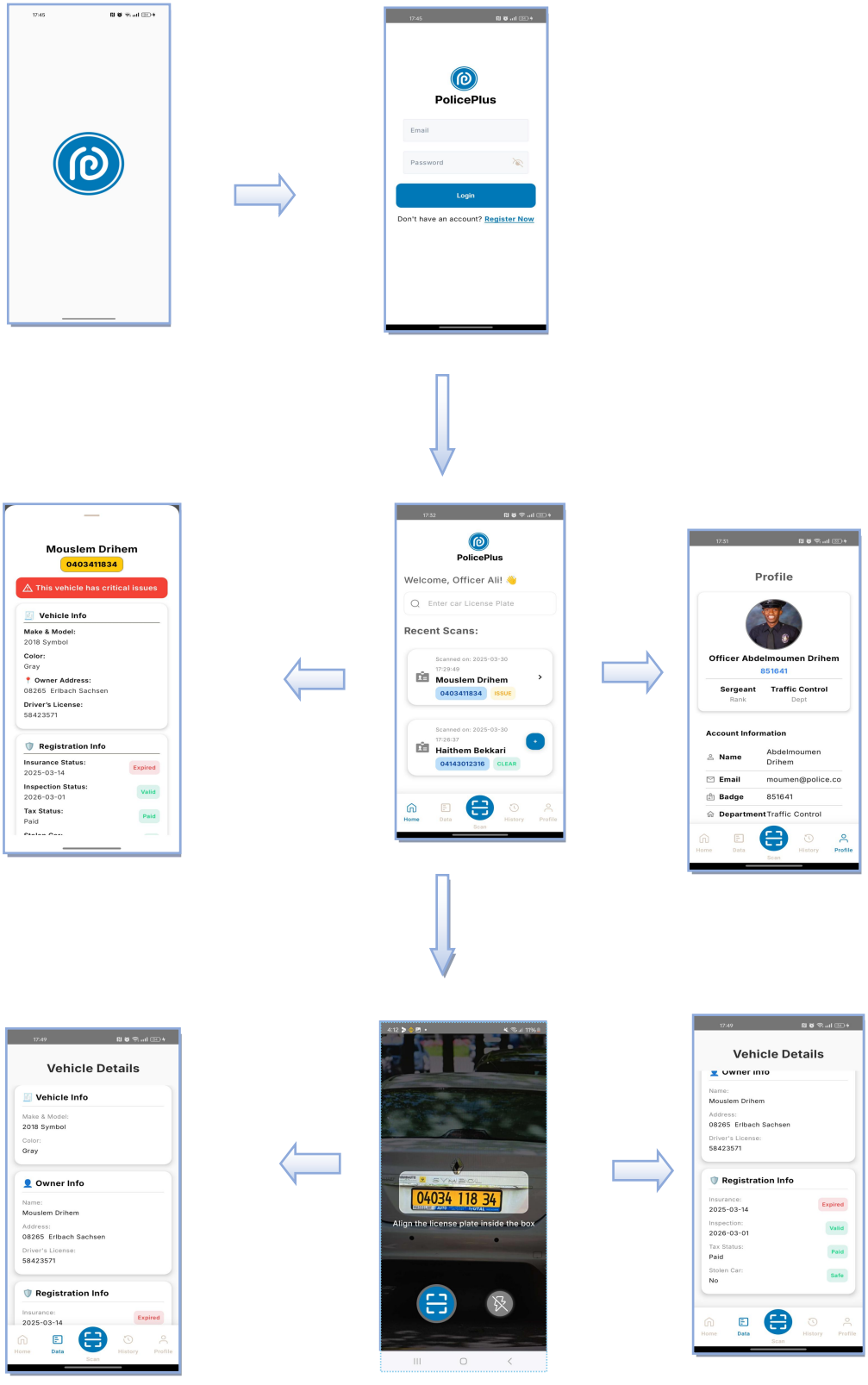


Figure III.2: General Presentation of the Parent Application

The application provides two login options either 'Police Officer' or 'User':

To begin, the officer is required to enter a 'Username' and 'Password' when launching the application. If no account exists, registration must be completed by providing a Badge Number, Rank, Department, and Email address.

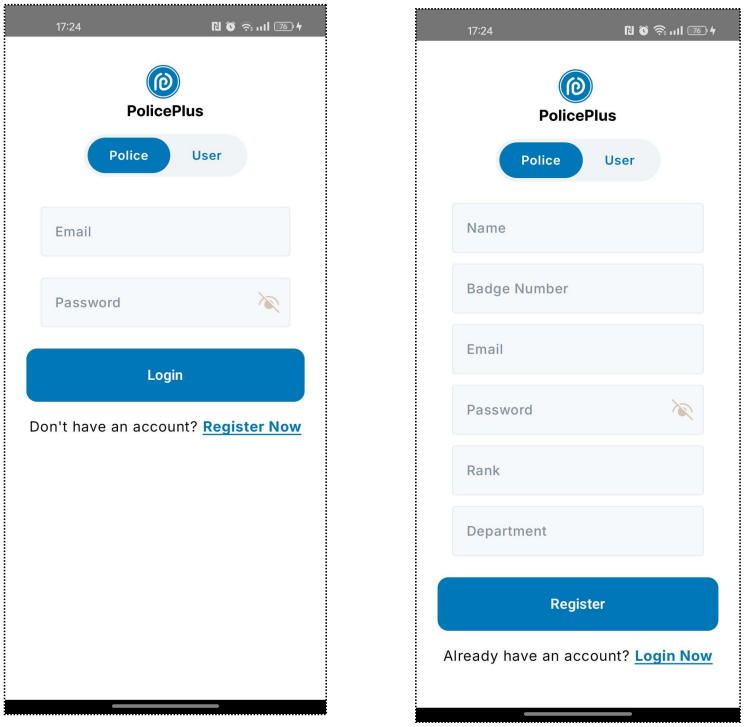


Figure III.3: Police Officer Login & Register page

a. «Main» Interface:

This is the main interface. It is displayed after verifying the 'Username' and 'Password' and shows the user's activities. A message will appear asking for permission to receive notifications from the application. The interface allows the user to choose from the options: 'Home,' 'Data,' 'Scan,' 'History,' and 'Profile'.

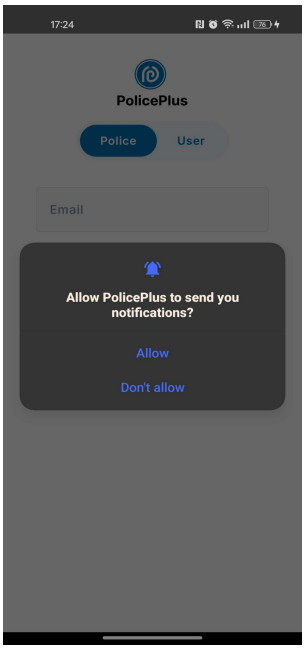


Figure III.4: permission to receive notifications

b. «Home» Interface:

Use the search bar for quick vehicle lookups by entering the vehicle number and clicking 'Search' to display the vehicle's details. Additionally, the last three scanned cars and the total number of scanned cars are displayed.

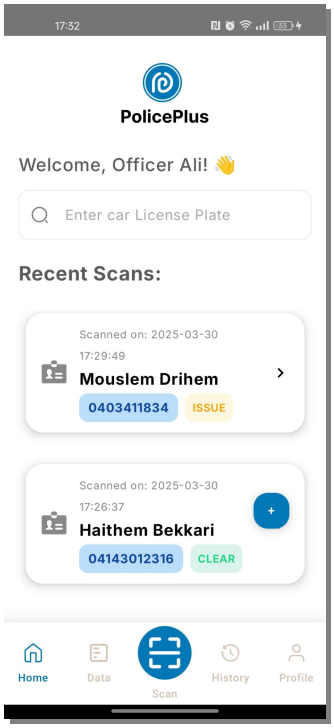


Figure III.5: Main menu 'Home'

c. «Scan» Interface:

License Plate Recognition and Data Retrieval: The officer can scan the vehicle number and confirm the scanned details.

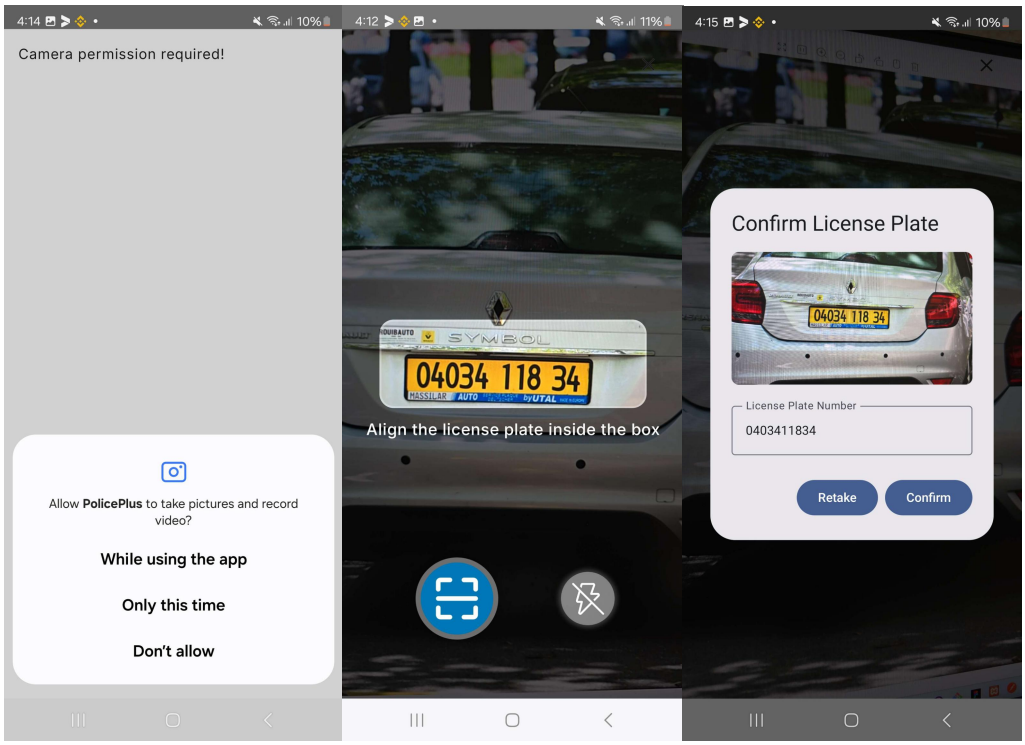


Figure III.6: Scan Interface

d. «Data» Interface:

Here, all information is displayed, including owner details, insurance start and end dates, stolen vehicle status, and any unpaid tickets.

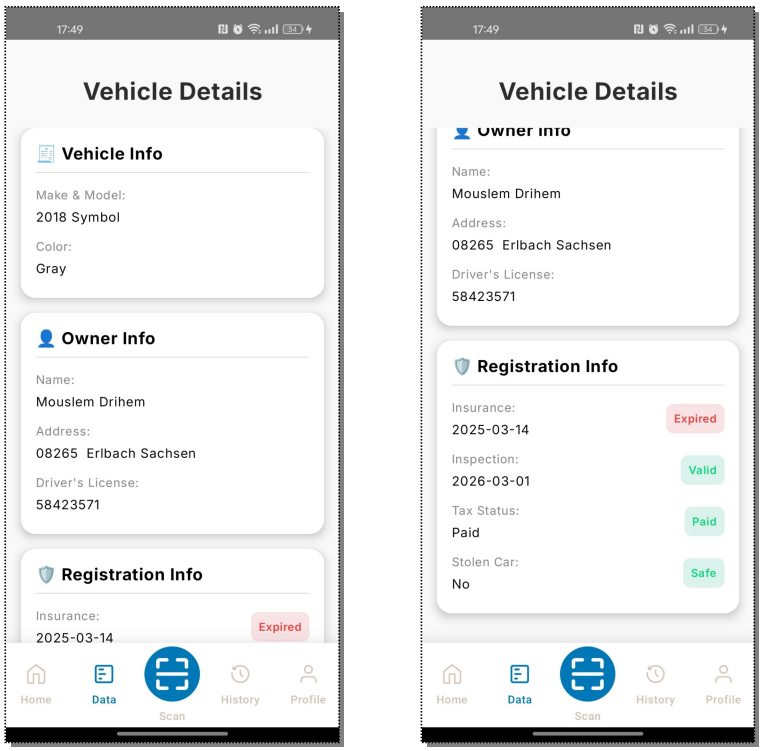


Figure III.7: Data Interface

e. «History» Interface:

The app allows officers to view previous scans with full details and filter records by selecting 'Stolen,' 'Issue,' or 'All.

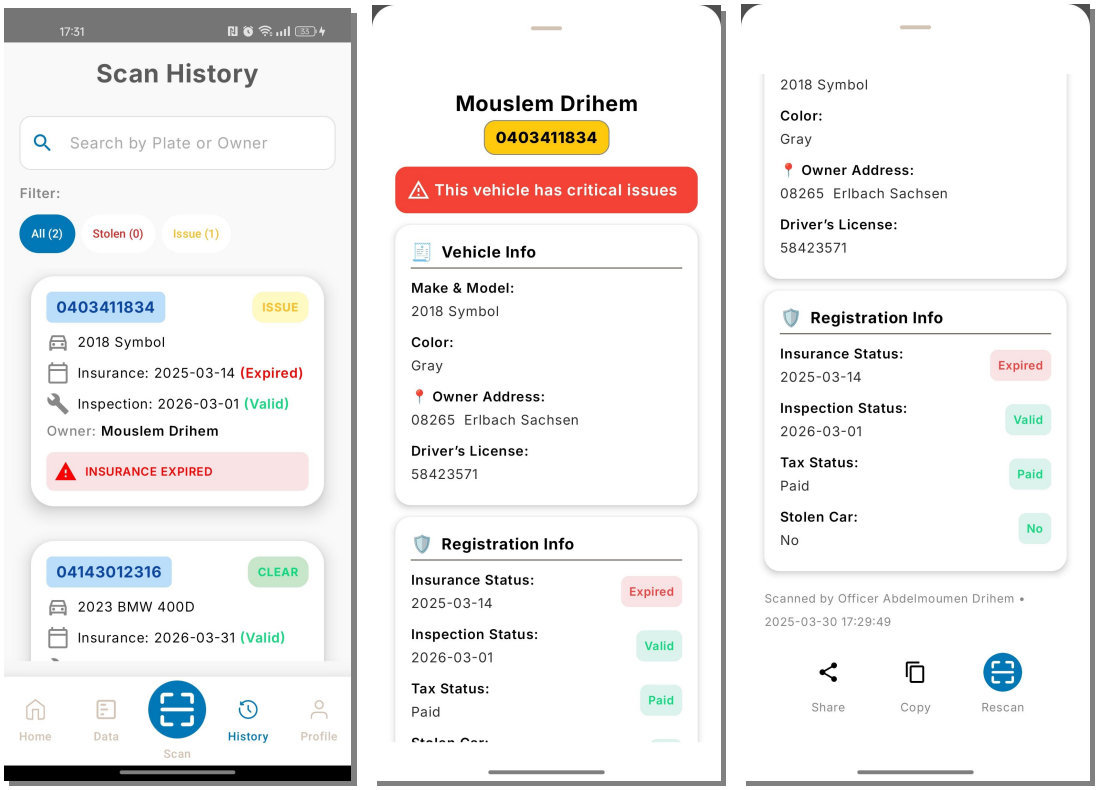


Figure III.8: History Interface

f. «Profile» Interface:

The officer can view account information and log out when needed.

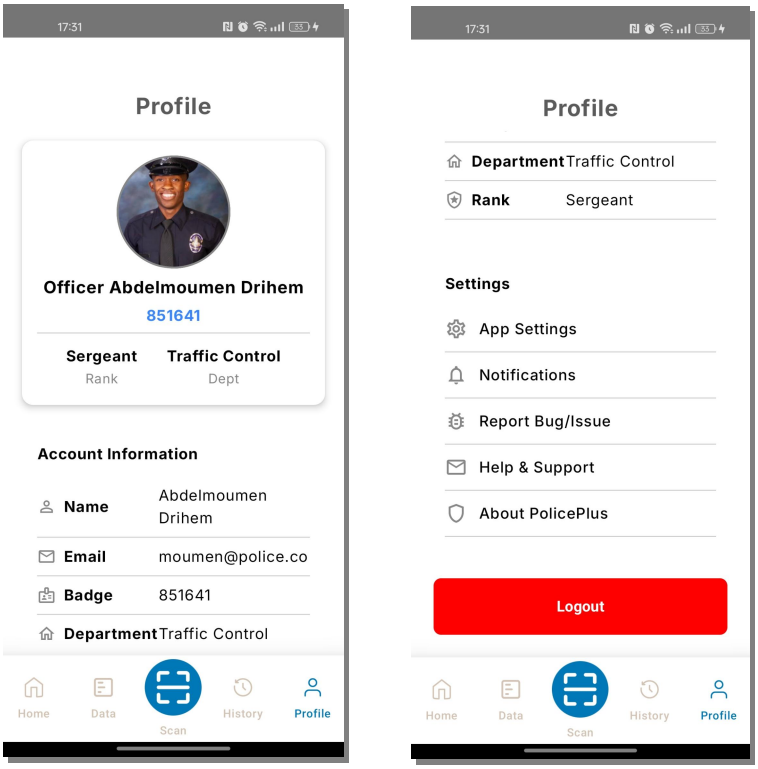


Figure III.9: Profile Interface

g. Ticket Interface:

The officer can issue a ticket.

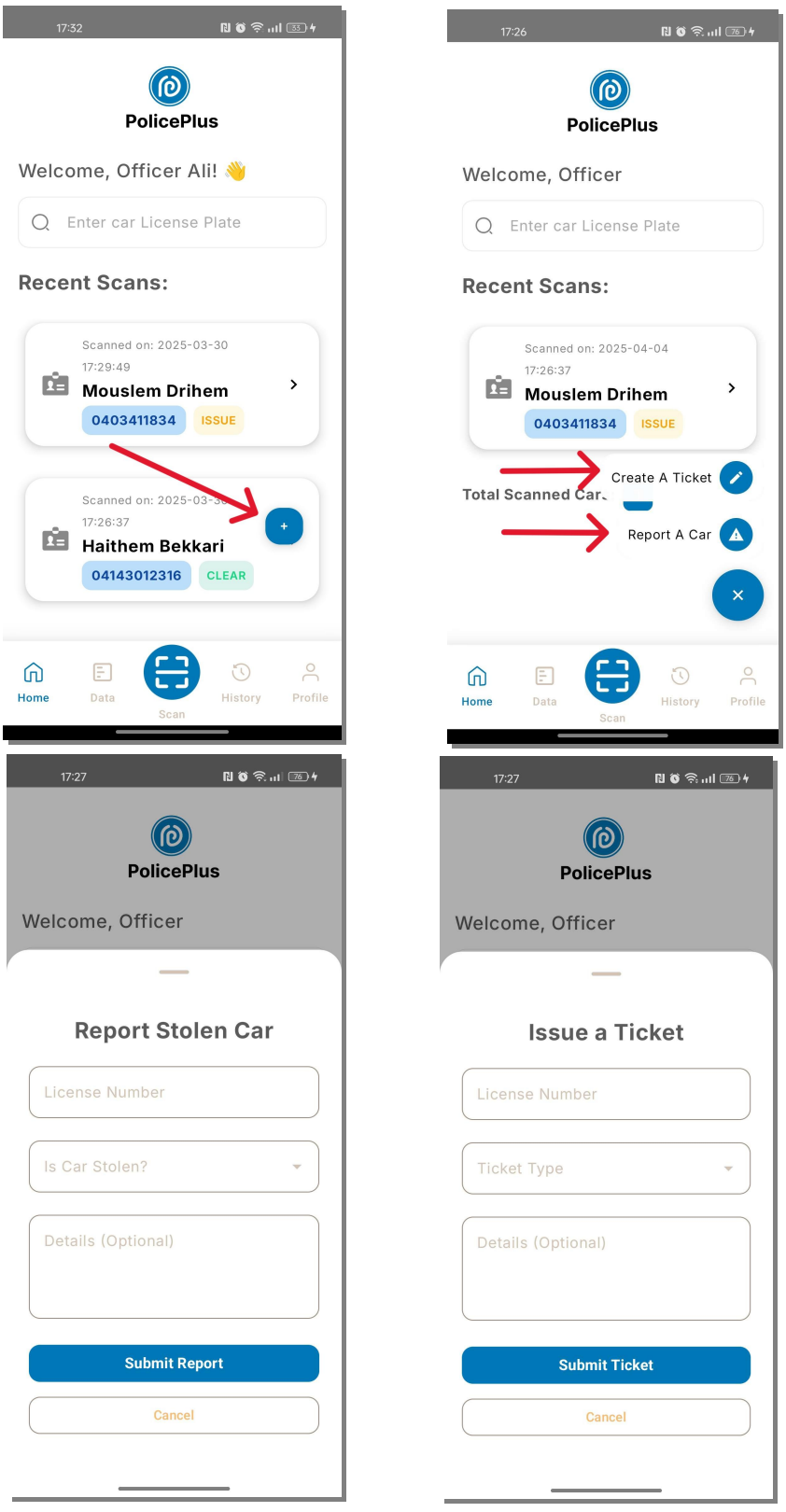


Figure III.10: issue a ticket(right), Report Stolen Car(left)

Alternatively, the user must enter their Email and Password. If they do not already have an account, they will need to register by providing their Name, Email address, Password, and Car License Plate.

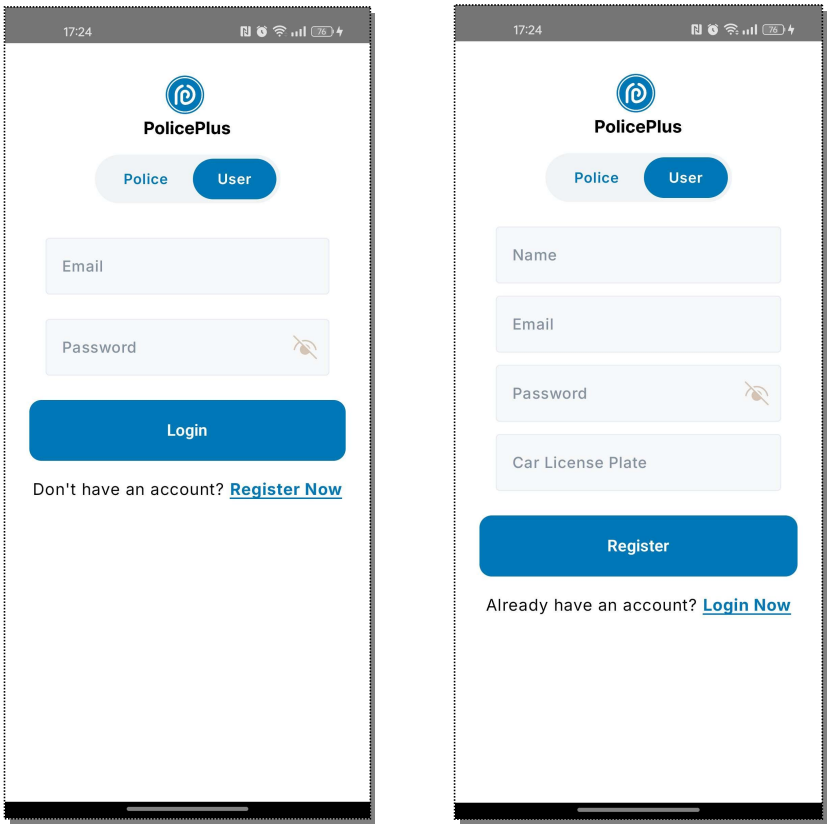


Figure III.11: User Login & Register page

h. «Main» Interface "User":

The main interface appears after verifying the user information. It displays user activities and prompts for notification permissions. Users can navigate to: Home and Profile.

i. «Home» Interface "User":

All car details are displayed here, including the owner, model, color, driving license, and registration information such as insurance, inspection, car registration, and tax.

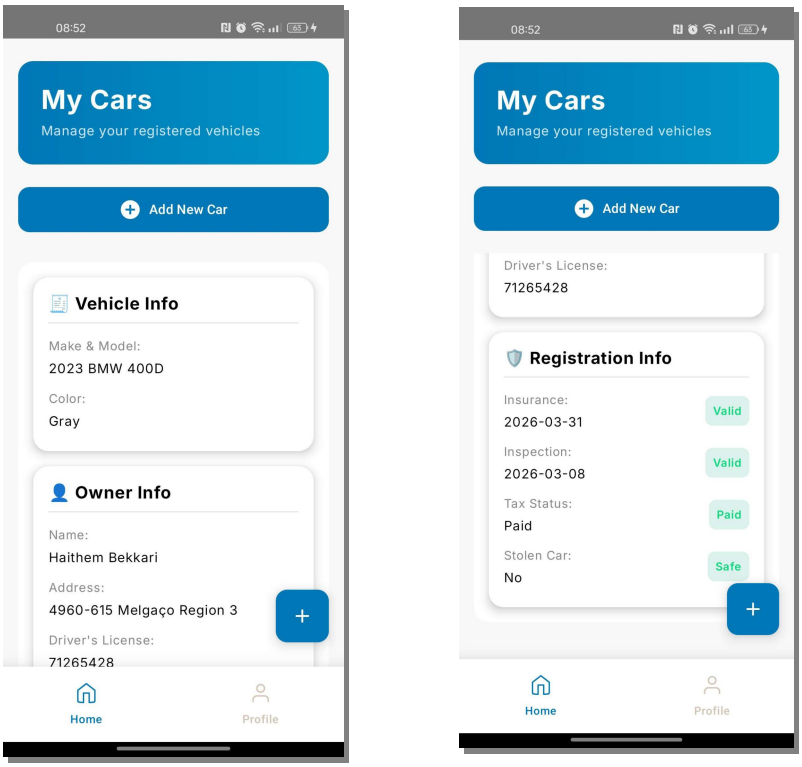


Figure III.12: Home page

j. Add new car Interface :

To add a new car, click the button and enter the license plate number.

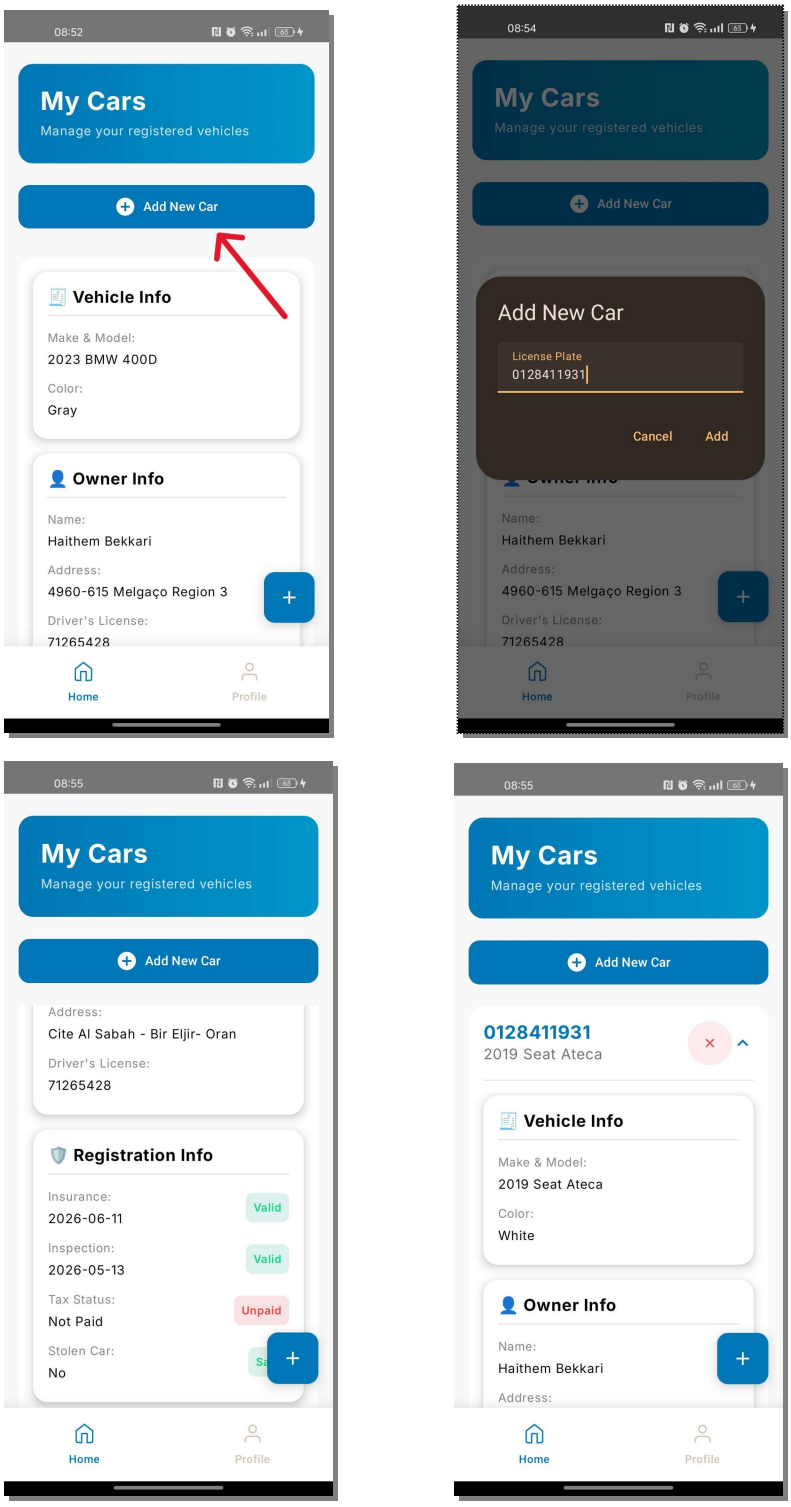


Figure III.13: Add new car

k. User profile :

The Profile page displays the user's account information, including their name, email address, and driver's license details.

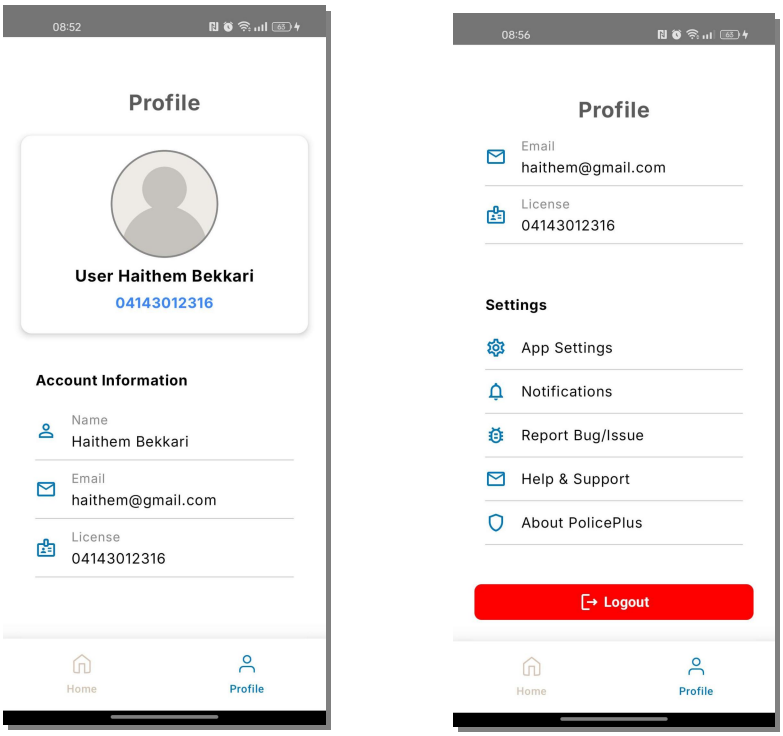


Figure III.14: User Profile

l. Report a car

The user can report a vehicle by providing the following information:

- License Number
- Indicate if the vehicle is stolen (Yes/No)
- Additional details (optional)

Then, click submit report.

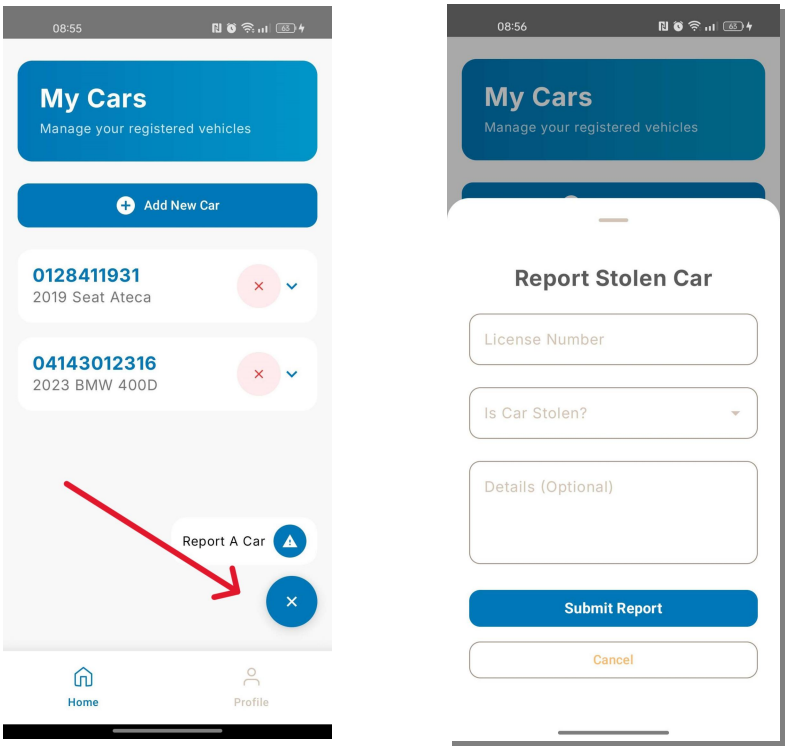


Figure III.15: Report a car

6. Conclusion

In this chapter, we presented the development environment and the main interfaces of our police application. These interfaces offer all the necessary features for police officers and other users to manage tasks efficiently, such as vehicle tracking, reporting, and accessing important information.

General conclusion

This project focuses on tracking vehicle statuses and assisting police officers in verifying vehicle-related information.

We have developed an application that enables law enforcement to efficiently access and manage vehicle data in real time.

The application allows police officers to check essential vehicle documents, including the registration certificate, driver's license, insurance details, technical inspection status, and tax receipts. It also facilitates quick identification of anomalies, such as stolen vehicles or expired documents. Additionally, officers can receive alerts and take immediate action when necessary.

For the development of our application, we used Android Studio, Adobe Figma, and MySQL.

We also plan to enhance the system with offline functionality and more security measures to improve efficiency and reliability.

List of sources and references

[1] : Room Database in Android(medium.com)
[2] : Meet Android Studio(developer.android.com)
[3] : Kotlin overview(developer.android.com)
[4] : MySQL(wikipedia)
[5] : Jetpack Compose(developer.android.com)
[6] : Understanding Dagger Hilt: Simplified Dependency Injection for Android(medium.com)
[7] : Retrofit in Android(medium.com)
[8] : Introduction to Node.js(nodejs.org)
[9] : What is Express.js? (medium.com)
[10] : Slow rendering(developer.android.com)
[11] : Clever cloud(clever-cloud.com)
[12] : What is Postman?(postman.com)
[13] : Figma(wikipedia)
[14] : About GitHub and Git (docs.github.com)