République Tunisienne

Ministère de l'Enseignement Supérieur

et de la Recherche Scientifique

École Supérieur Privée d'ingénierie et de technologie

TEK-UP

Rapport de Projet de Fin d'Études

Présenté en vue de l'obtention du

Diplôme National d'Ingénieur en Informatique

Spécialité : Genie Logiciel

Réalisé par

BOUSSAIDI Bayrem

Système de Location en Ligne avec Interaction Multi-Utilisateurs et Gestion Dynamique des Réservations

Encadrant professionnel :	**M. ZELLIT Mootaz**	Ingénieur Informatique
Encadrant académique :	**Mme. JALEL Sawsen**	Poste

Année Universitaire 2024 - 2025

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant professionnel, **M. XXXXXXX**

Signature et cachet

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant académique, **M. YYYYYY**

Signature

Thanks

I dedicate this work :

To my dear mother Zahida,
Thank you for your unconditional love and unwavering support that have guided me throughout this journey.
Your love has been the light that showed me the way, and I am forever grateful to you.

To my dear father Ali,
Your strength, hard work, and encouragement have always inspired me to give my best.
You have been a model of perseverance and determination.

To my brother Ahmed and my sister Tasnim,
Your presence and support have been invaluable to me.
Thank you for all the moments we have shared and for your precious care.

This work is dedicated to all of you, with all my gratitude and affection.

– Bayrem Boussaidi –

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

API = Application Programming Interface

CI/CD = Continuous Integration / Continuous Deployment

DB = Database

GLSI = Software Engineering and Information Systems

K8s = Kubernetes

PDF = Portable Document Format

QR = Quick Response (Code)

UI = User Interface

UX = User Experience

General Introduction

The development of our Car Rental Comparator Platform aims to address the growing demand for digitalization and optimization in the car rental sector. Our application offers an integrated solution for agencies to list, manage, and rent out their vehicles in a dynamic and efficient ecosystem. It enables agencies to maximize the visibility and value of their services while minimizing administrative overhead.

Our platform stands out by facilitating seamless communication between agencies and administrators, providing a space where agencies can create, update, and remove their listings effortlessly. Agencies are now directly responsible for managing rental requests for their listed vehicles — they can approve or decline requests submitted by customers. Upon agency approval, a secure payment link is sent to the customer's email, allowing them to finalize the transaction online. Once payment is confirmed, the platform automatically generates and sends a PDF rental contract to the customer's email, including the rental details and a uniquely generated QR code.

Additionally, the platform features a blogging space for special offers and events, enhancing customer engagement.

To enrich the customer experience, we introduced a subscription system for premium clients, keeping them updated with the latest offers and news. Clients can easily reserve cars online, check real-time vehicle availability, and enjoy a streamlined booking process.

Our application is developed using **Angular** for the front-end, **Spring Boot** for the back-end, and **MySQL** for the database. The entire infrastructure is containerized using **Docker** and orchestrated with **Docker Compose**, ensuring consistency across development and production environments.

The platform is hosted on a virtual server provided by **AWS EC2**, offering high availability and scalability. Infrastructure provisioning — including instance creation, networking, security groups, and database setup — is fully automated using **Terraform**, enabling reproducible and secure deployments. To streamline development and deployment cycles, we have set up a complete **CI/CD pipeline** using **GitLab**, which automates code integration, Docker image builds, and deployment to the AWS environment.

This report will detail the design and development approaches adopted, the technologies and tools utilized, and the potential impacts and advantages of this platform for its users. We will also discuss prospects for similar future projects. The report will be organized as follows :

— In the first chapter, we will introduce the project framework and its general context, followed by a state-of-the-art study analyzing three similar applications. The project management methodology adopted will also be presented.

— The second chapter will focus on the analysis and specification of project requirements, identifying actors, functional and non-functional needs, and presenting use case and class diagrams, along with release planning, application architecture, and the working environment.

— The third chapter will detail the first release of the platform, including three sprints : profile and car listing management, advertisement management, and request handling, outlining the implementation stages and features delivered.

— The fourth chapter will present the second release, which includes the management of rental requests, rental contracts, and customer complaints, thus enriching the platform's functionalities and enhancing the overall user experience.

— The fifth chapter will cover the realization of the third release, integrating features for maintenance request management, payment invoice generation, and the establishment of the CI/CD pipeline, ensuring continuous delivery and service improvement.

Chapitre 1

Project Context

Plan

1.1 Introduction

This chapter presents the general context of the project. It introduces the host organization, **Myloc Agency**, analyzes existing methods, and describes the proposed solution. Additionally, it outlines the project architecture, design approaches, management methodology, and development environment.

1.2 Presentation of the Host Organization : Myloc Agency

This section provides an overview of the host organization, its mission, activities, and products, highlighting its innovative approach and global impact.

1.2.1 Host Organization

Myloc Agency leverages expertise in car rental services to offer customized solutions that provide reliable and affordable transportation options for clients. Founded by Amine Ben Chaabane, it consists of a passionate and dedicated team of skilled professionals who work closely with clients to deliver transportation solutions tailored to their specific needs.

1.2.2 Activities and Products

As part of this project, I developed a full-featured car rental web application that supports three main user roles : Admin, Agencies, and Customers, with additional functionalities for visitors and followers. The application includes a secure, role-based system powered by Keycloak for authentication and route protection. It was built using Spring Boot (backend), Angular (frontend), FastAPI (chatbot backend), and a separate payment microservice, all containerized with Docker, orchestrated with Docker Compose, deployed on Kubernetes (K8s), and integrated with GitLab CI/CD pipelines for continuous deployment.

1.3 Project Frame

This section delves into the framework of the project, starting with an analysis of existing systems to identify their strengths and weaknesses. By understanding gaps and opportunities in current solutions, we propose an innovative approach that addresses these shortcomings and enhances the overall user experience.

1.3.1 Study of Existing Systems

This section explores existing car rental platforms to identify their functionalities, strengths, and limitations. The aim is to evaluate how current systems handle administrative, agency, and customer interactions, and to highlight areas where improvements can be made.

— **Traditional Car Rental Websites** : Many companies offer web platforms where users can search and book vehicles. These platforms often lack real-time availability checks, multi-role management, or automated document generation. Interaction between customers and agencies is usually limited or entirely missing.

— **Aggregator Platforms (e.g., Kayak, Rentalcars)** : These platforms centralize listings from multiple rental agencies, allowing users to compare prices and availability. While convenient for users, they do not provide backend control for agencies or real-time request management. Personalized communication is also limited.

— **Custom Enterprise Systems** : Some large agencies have internal systems for managing cars, customers, and payments. These solutions are often closed-source, expensive, and not scalable for small or medium-sized agencies. Many lack modularity, chat features, or automated workflows like PDF contract generation and QR code integration.

— **Limitations Identified** :

  — Lack of real-time communication between agencies and customers.

  — No integrated notification or email system for status updates.

  — Limited role-based access (admin, agency, customer).

  — Absence of automation in rental confirmation workflows (e.g., contract generation, online payment).

  — Poor support for multi-agency environments within a single system.

1.3.2 Gaps and Opportunities

This section identifies the key limitations in existing car rental systems and highlights opportunities for technical and functional improvement.

— **Lack of Multi-Role Interaction** : Most platforms do not support smooth interaction between different types of users. A robust role-based system with protected routes and custom dashboards is necessary.

— **Insufficient Real-Time Communication** : Many systems lack instant messaging features. Adding a built-in chat system can improve customer service and speed up booking confirmation.

— **Limited Automation** : Automated PDF contract generation, QR code creation, and

email notifications are rarely offered. These features streamline workflows and enhance user satisfaction.

— **Weak Integration of Notifications and Updates** : Timely updates for bookings, new cars, and promotions are missing. An integrated notification system (email + in-app) is essential.

— **Lack of Dynamic Availability Checking** : Users can book cars without real-time validation, creating double bookings. Implementing date-based availability checks resolves this.

— **Limited Customization for Agencies** : Smaller agencies often lack control over fleet management or platform display. Individual dashboards with CRUD capabilities are an opportunity.

— **Underuse of Modern Deployment Practices** : Few platforms use CI/CD pipelines, containerization, or Kubernetes, making scaling and maintenance difficult. Leveraging Docker, Docker Compose, K8s, and GitLab CI/CD ensures stability and faster updates.

Gap	Exists in current platforms
Multi-role management with custom permissions	LIMITED
Real-time messaging between users	RARE
Automatic notifications and email updates	LIMITED
Dynamic car availability checking	NO
PDF contracts with QR code generation	NO
Online payment integration with validation flow	LIMITED
Scalable deployment (CI/CD, Docker, K8s)	NO

Tableau 1.1 : Identified Gaps in Existing Car Rental Platforms

1.3.3 Proposed Solution

The proposed solution addresses the gaps identified in current rental platforms :

— **Role-Based System with Secured Access** : Supports multiple roles (admin, agency, customer, visitor) with dashboards and protected routes via Keycloak.

— **Real-Time Communication** : Built-in chat for agencies and customers.

— **Automated Notification and Email System** : Sends automatic updates for bookings, new cars, and account activities.

— **Contract Generation with QR Code** : PDF rental agreements generated and sent via email after successful payment.

— **Smart Availability Checker** : Validates car availability before confirming bookings.

— **Modern Infrastructure and Deployment** : Deployed using Docker Compose, Kubernetes, and GitLab CI/CD for scalability and maintainability.

— **Integrated Chatbot Assistant** : FastAPI-based chatbot using OpenAI for real-time user

assistance.

1.4 Project Architectures and Design Approaches

1.4.1 System Architecture : Microservices with Three-Tier Logical Structure

The system follows a modern microservices architecture with clear separation of concerns across multiple tiers. The architecture ensures scalability, maintainability, and security through containerized deployment and CI/CD practices.

— **Presentation Layer** : Angular frontend handling UI and user interactions, communicating with backend via REST APIs.

— **Business/Application Layer** : Spring Boot and FastAPI microservices for booking, chat, role-based access, notifications, and payment workflows.

— **Data Layer** : MySQL relational database ensuring data integrity and advanced queries.

1.4.2 Design Pattern : RESTful Service-Oriented Architecture (SOA)

— Stateless interactions, resource-oriented endpoints, loose coupling, reusability, and clear separation of concerns.

1.4.3 Design Pattern : MVC for Payment Service

— Model : Data entities and payment logic.

— View : JSON responses and webhook callbacks.

— Controller : API request handling, transaction validation, and workflow triggering (PDF/QR).

1.4.4 FastAPI Chatbot Microservice

— Lightweight RESTful endpoints for AI-powered chatbot interactions.

— Modular logic for maintenance and scaling.

— Containerized via Docker, deployed with Kubernetes.

1.5 Project Management and Design Methodology

1.5.1 Agile Methodology

Development is organized in small iterative cycles, encouraging feedback, team collaboration, and continuous improvement. Scrum is used for sprint planning, daily stand-ups, and iterative delivery.

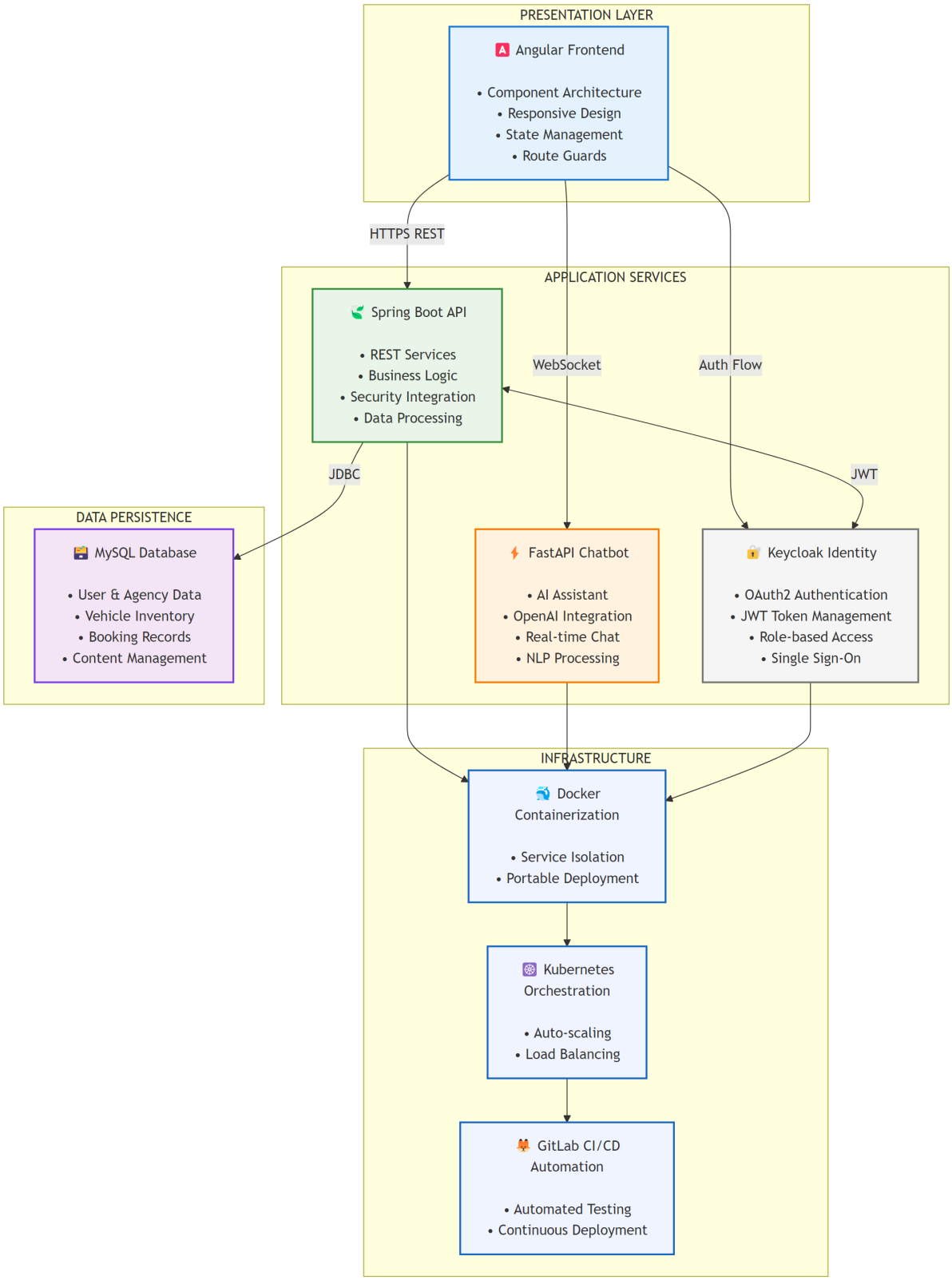

Figure 1.1 : System Architecture Overview - Myloc Agency Car Rental Platform

Component	Specification
Processor	Intel Core i7
RAM	16 GB DDR4
Storage	512 GB SSD
Operating System	Windows 10 + WSL2
Display	15.6-inch Full HD

Tableau 1.2 : Development Machine Specifications

1.5.2 Development Environment

1.5.2.1 Hardware Environment

1.5.2.2 Software Environment

— Programming Languages : Java, Python, TypeScript, HTML/CSS/JS.

— Frameworks : Spring Boot, FastAPI, Angular, Keycloak.

— Containerization : Docker, Docker Compose.

— Orchestration : Kubernetes (K8s).

— CI/CD : GitLab CI/CD.

— Database : MySQL.

— Testing : Postman, XAMPP.

— IDE : Visual Studio Code.

— Version Control : Git, GitLab.

— Project Management : Scrum, Trello/GitLab Issues.

1.5.2.3 Deployment Tools

— Docker Compose for local service orchestration.

— Kubernetes (K8s) for scalable microservice deployment.

1.6 Problematic

The main challenge addressed by this project is the absence of a centralized and automated car rental platform capable of supporting multiple user roles, real-time communication, and dynamic availability validation. Existing solutions are fragmented, lack interaction between agencies and customers, and rely on manual processes that hinder efficiency and user satisfaction.

1.7 Conclusion

The chosen stack and practices enabled efficient development of a scalable, secure, and automated car rental platform with real-time messaging, QR-based contracts, automated notifications, and AI chatbot assistance. This project highlights the benefits of combining modern web technologies, microservice architecture, and agile practices to create a multi-role, interactive, and robust system.

Chapitre 2

Needs Analysis and Specification

Plan

2.1 Introduction

This chapter formally defines and analyzes the functional and non-functional needs of the car rental platform. By identifying limitations in existing systems, this section translates user expectations into specific system requirements, focusing on secure, real-time, and scalable services for car rentals, including role-based interactions, authentication, booking workflows, communication, and deployment. Following the context and problem analysis outlined in Chapter 1, this chapter ensures that the platform meets real user needs efficiently and securely.

2.2 Actors Identification

The platform supports the following primary actors :

— **Administrator** : Manages agencies, blogs, customers, and system security. Has access to all system functionalities.

— **Agency** : Manages cars, responds to rental requests, communicates with the admin, and handles customer interactions.

— **Customer** : Can view public content, submit rental requests, interact with agencies, and track requests post-login.

— **Visitor** : Unauthenticated user who can view public content and access the chatbot.

— **Chatbot (System)** : AI-powered assistant via a FastAPI service connected to ChatGPT, available for visitors and registered users.

2.3 Needs Identification

2.3.1 Functional Needs

— **Customers :**

  — Browse available cars and blogs without logging in.

  — Register, log in securely, and manage their profiles.

  — Search and filter cars based on availability and specifications.

  — Submit rental requests for cars with date availability checked.

  — Receive email notifications on rental request status (accepted/declined).

  — Access notifications and view rental request history.

  — Delete rental requests that are declined or not confirmed.

  — Comment and provide reviews on cars and blog posts.

  — Receive updates via email when new cars are added or blogs published.

  - Contact admin directly via email within the app.
  - Interact with the chatbot for support and inquiries.
- **Agencies :**
  - Register, log in securely, and manage agency profiles.
  - Add, edit, or delete cars and their features (name, model, type, photos).
  - Accept or decline rental requests from customers.
  - Send automated email responses to customers based on request status.
  - Communicate with admin via instant chat within the app.
  - Manage car availability and scheduling.
  - Receive notifications of new rental requests.
- **Administrators :**
  - Manage agency accounts : add, edit, and delete agencies.
  - Manage customers : add, edit, and delete customer accounts.
  - Manage blog posts : add, edit, and delete blog content.
  - Oversee platform-wide notifications and communications.
  - Monitor system health and enforce security via secured route access (using Keycloak and AuthGuard).
  - Engage in chat with agencies for operational coordination.
  - Configure system settings and roles for access control.
- **System-wide Functionalities :**
  - Secure authentication and role-based authorization via Keycloak.
  - Instant messaging system enabling admin-agency communication.
  - Email notification system for updates, rental request status, and confirmations.
  - PDF generation with QR codes for confirmed rental contracts.
  - Integration of chatbot powered by an external Flask API for user assistance.
  - Deployment automation using GitLab CI/CD pipelines, Docker Compose, and Kubernetes.
  - Real-time availability checking for cars during rental request submission.

2.3.2 Non-Functional Needs

- **Performance :**
  - The system must respond to user actions (e.g., browsing cars, submitting rental requests) within 2 seconds under normal load.
  - Rental availability checks must provide real-time feedback during request submission.
  - Email notifications and PDF generation should be triggered within 2 minutes after any

change in request status.

- **Security :**
  - Authentication and authorization must be enforced using Keycloak, with role-based access control (Admin, Agency, Customer).
  - All sensitive data (e.g., passwords, personal information) must be securely stored and transmitted using encryption protocols (HTTPS, password hashing).
  - All API endpoints must be protected from unauthorized access using AuthGuard and secure access control mechanisms.
  - The application must defend against common security threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- **Usability :**
  - The user interface must be intuitive, user-friendly, and responsive across desktop, tablet, and mobile devices.
  - Customers must be able to easily browse, search for cars, submit rental requests, and track their status.
  - Agencies and administrators should have dedicated dashboards for managing cars, requests, blogs, and communication.
- **Reliability and Availability :**
  - The system should ensure 99.9% uptime, with robust error handling and recovery mechanisms.
  - All rental and user data must be reliably saved and backed up regularly.
- **Maintainability and Extensibility :**
  - The codebase must follow clean architecture principles and separation of concerns to facilitate future updates.
  - Continuous integration and deployment (CI/CD) pipelines using GitLab must support automated testing and deployment.
  - Comprehensive documentation should be maintained for the API, deployment process, and system usage.
- **Scalability :**
  - The system must support increased user traffic and rental activity without performance degradation.
  - Kubernetes-based deployment must allow for horizontal scaling of services as demand grows.
- **Interoperability :**
  - The system must integrate seamlessly with external services, including the Flask-based

chatbot API and payment gateways.

— Email services must be compatible with SMTP or third-party providers to ensure reliable communication.

2.4 System Design

2.4.1 Global Use Case Diagram

The global use case diagram represents the overall interactions between primary actors (Customer, Agency, Administrator, Visitor) and the car rental platform. It highlights the key functionalities accessible to each actor, offering a high-level overview of the system's capabilities.

2.4.1.1 Customer Use Case Diagram

The customer use case diagram illustrates the primary interactions between customers and the car rental platform. It shows how registered customers can manage their profiles, search and book vehicles, interact with the AI chatbot, and engage with the platform's content system.

2.4.1.2 Agency Use Case Diagram

The agency use case diagram demonstrates how car rental agencies interact with the platform to manage their fleet, handle customer bookings, and maintain their business operations through the comprehensive agency dashboard.

2.4.1.3 Administrator Use Case Diagram

The administrator use case diagram showcases the comprehensive system management capabilities available to platform administrators, including user management, content moderation, financial oversight, and system maintenance operations.

2.4.2 Global Class Diagram

2.5 Specification

2.5.1 Role Allocation

Scrum Master (SM)

— **Responsibility :** Facilitates Scrum ceremonies, removes impediments, and ensures Agile principles are followed.

— **Interaction :** Acts as a servant leader between Product Owner and development team,

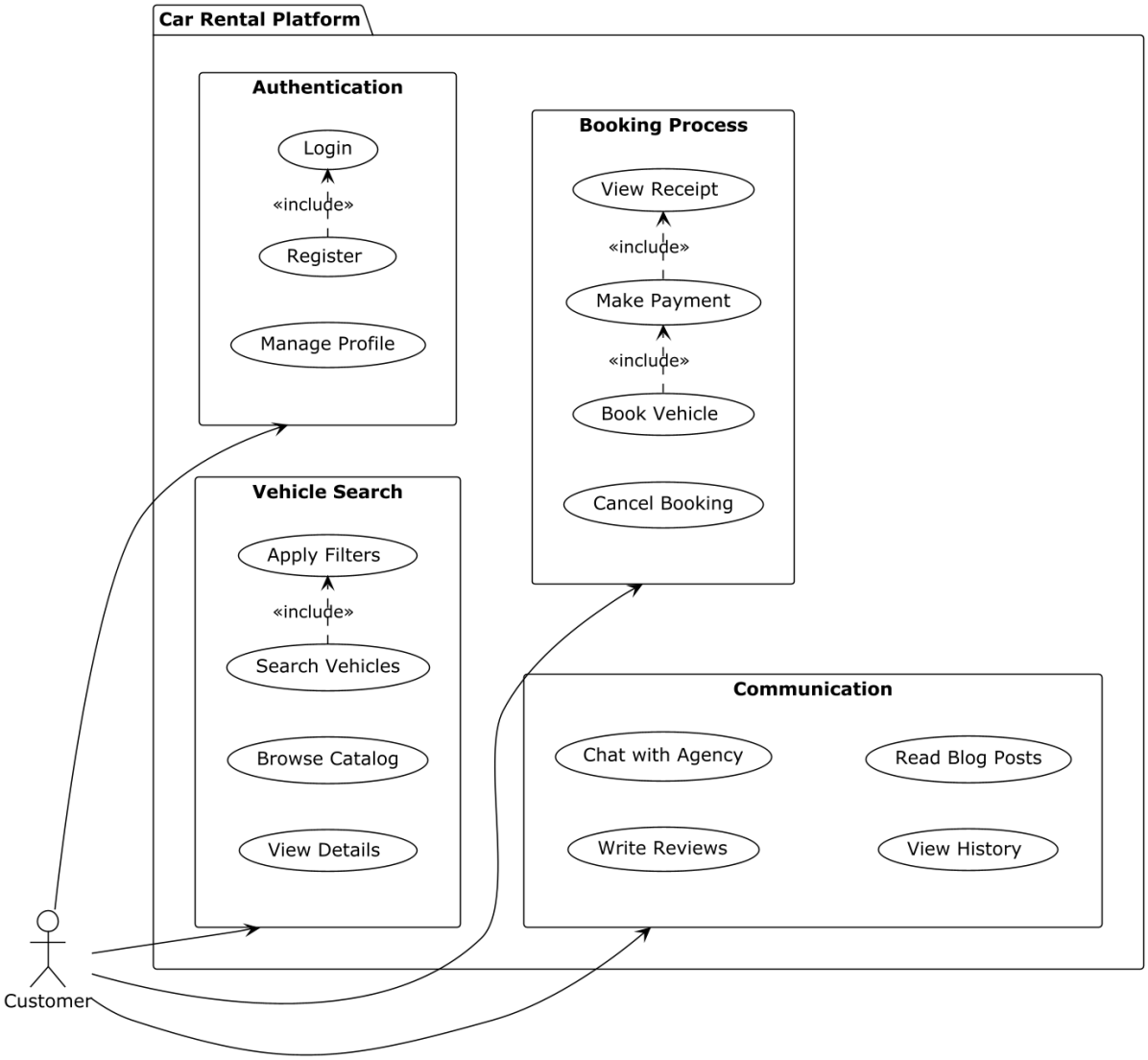

Figure 2.1 : Use Case Diagram for Customer - Myloc Agency Platform

maintaining productivity and team alignment.

Development Team (DT)

— **Responsibility :** Design, implement, test, and deliver functional features of the platform within each sprint.

— **Interaction :** Collaborate with Scrum Master and Product Owner to select tasks, implement solutions, validate outcomes, and deliver high-quality increments of the product.

2.5.2 Scrum Planning

— **Sprint Goal Definition :** Each sprint starts with a clearly defined goal contributing to project progress.

— **Backlog Refinement :** Regular review and prioritization of the product backlog.

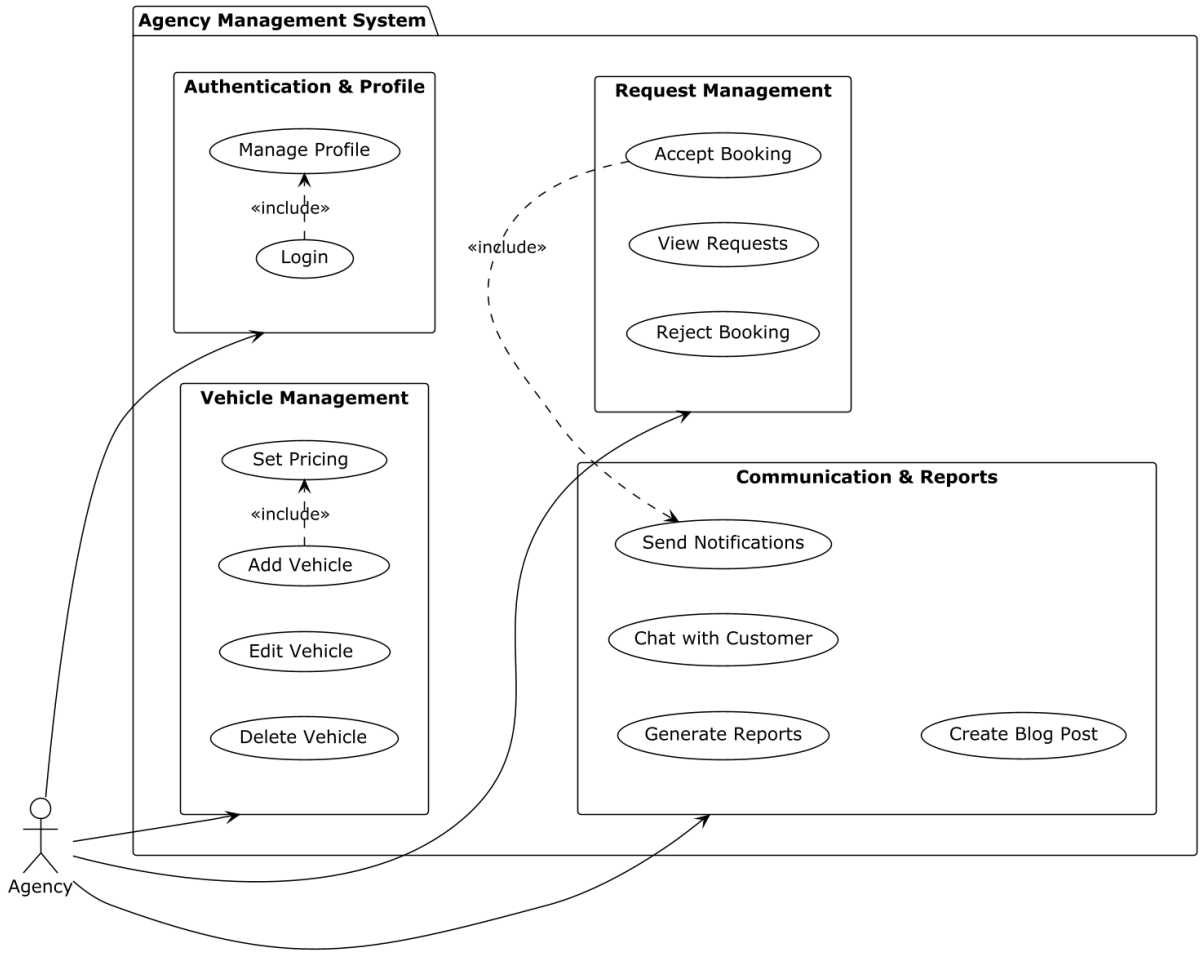

Figure 2.2 : Use Case Diagram for Agency - Fleet and Booking Management

— **Sprint Backlog Creation :** Select prioritized tasks and break them into actionable items.

— **Task Estimation :** Estimate tasks using story points or time-based measures.

— **Daily Stand-ups :** Short meetings to share progress, blockers, and daily goals.

— **Sprint Review and Retrospective :** Review completed features and reflect internally on improvements.

2.5.3 Sprint Planning Overview

	ID		
	1		Platform
	2		
	3		
	Agency)		
	4		
	5		

	ID
	6
	7
	8
	9
	10

Tableau 2.1 : Sprint Planning Overview

2.5.4 Conclusion

In this chapter, we explored the functional and non-functional requirements for the car rental platform. We identified the different user roles (Admin, Agency, Customer, Visitor), their needs, and outlined the system features accordingly. We also established the development methodology using Scrum, assigned team roles, and laid out the sprint plan and product backlog. This structured specification ensures the platform is developed efficiently, securely, and in alignment with user needs.

In the next chapter, we will focus on the **first sprint**, dedicated to platform setup, secure authentication using Keycloak, and user role management.

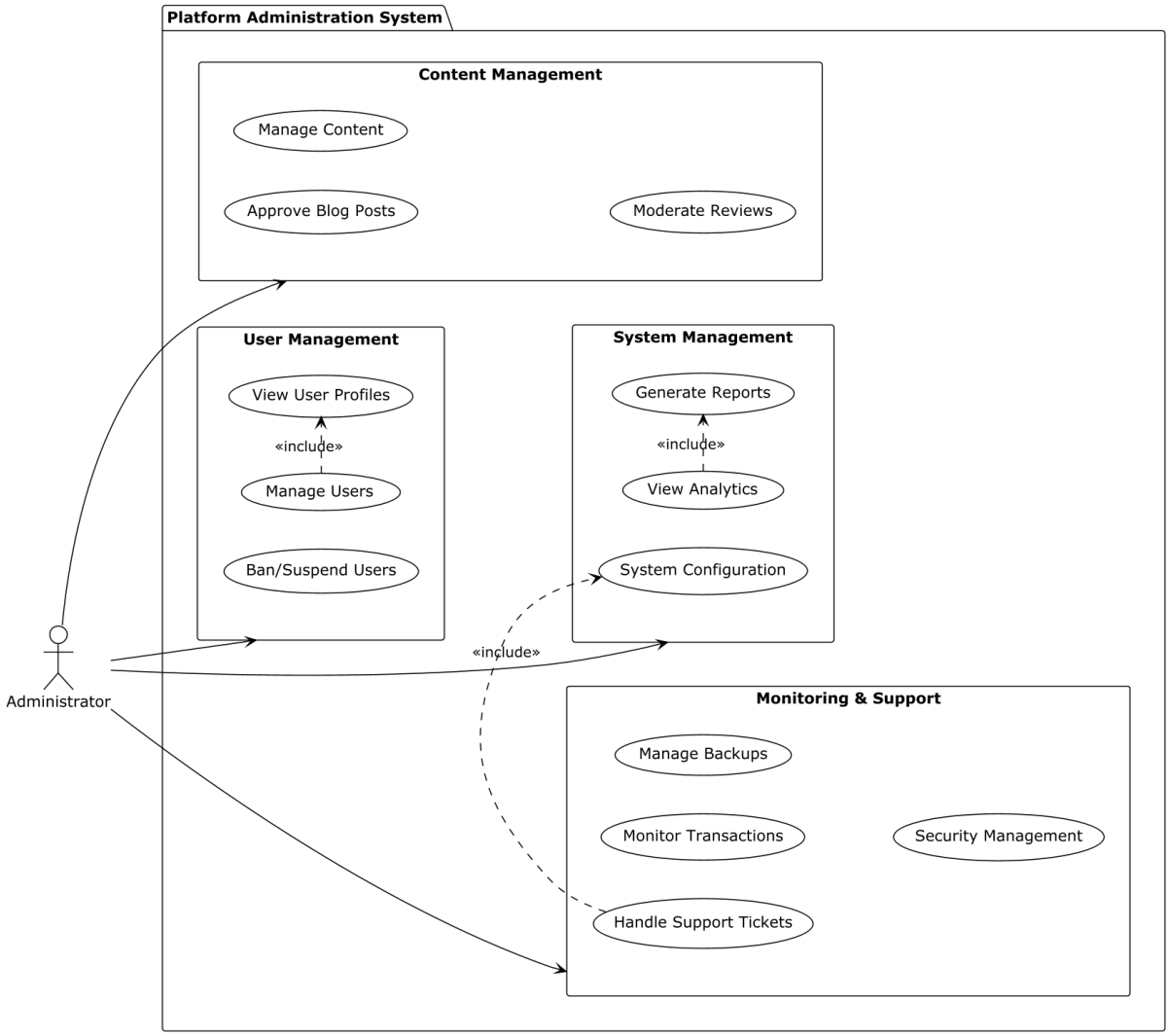

Figure 2.3 : Use Case Diagram for Administrator - System Management

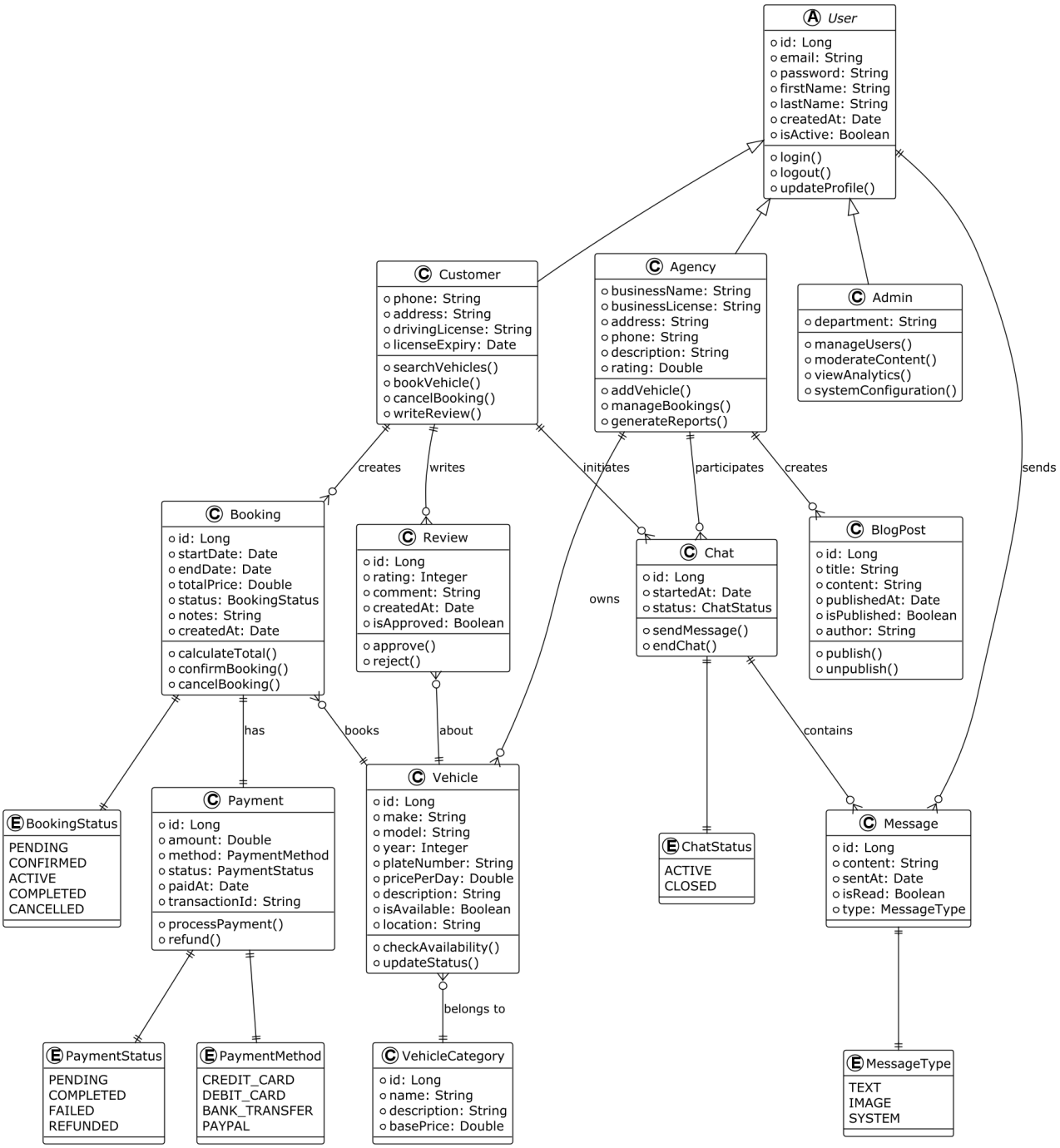

Figure 2.4 : Global Class Diagram Representing Car Rental Platform Architecture

Chapitre 3

System Development and Implementation

Plan

3.1 Introduction

This chapter details the development environment, system architecture, database design, and implementation strategies used for the Car Rental Platform. It also explains how functional requirements are translated into working modules and how non-functional requirements such as security, scalability, and real-time communication are implemented.

3.2 Development Environment

- **Frontend :** Angular 16, Node.js 20
- **Backend :** Spring Boot 3.x, Java 17 (REST API)
- **Database :** MySQL 8
- **Authentication :** Keycloak (Docker container)
- **Chatbot :** Flask API (Python) integrated with ChatGPT
- **Containerization :** Docker Compose
- **Orchestration :** Kubernetes (Minikube)
- **IDE :** Visual Studio Code
- **API Testing :** Postman
- **Local DB :** XAMPP

3.3 Database Design

The system uses MySQL as its relational database. Key tables include :

3.3.1 Agence Table

```
CREATE TABLE agence (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    agency_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(64) NOT NULL,
    photo LONGTEXT,
    phone_number VARCHAR(20) NOT NULL,
    city VARCHAR(100)
);
```

3.3.2 Blog Table

```
CREATE TABLE blog (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255),
    img_url VARCHAR(255),
    author VARCHAR(255),
    date VARCHAR(255),
    time VARCHAR(255),
    description VARCHAR(1024),
    quote VARCHAR(255)
);
```

3.3.3 Booking Table

```
CREATE TABLE booking (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255),
    user_email VARCHAR(255),
    phone VARCHAR(255),
    description VARCHAR(255),
    start_date DATE,
    end_date DATE,
    price DOUBLE,
    voiture_id BIGINT,
    nbr_jrs INT,
    booking_status VARCHAR(255),
    pickup_location VARCHAR(255),
    dropoff_location VARCHAR(255),
    car_name VARCHAR(255),
    agence TEXT
);
```

3.3.4 Other Tables

Additional tables include `customers`, `roles`, `notifications`, `followers`, and `chat_messages` to manage users, roles, communications, and subscription updates.

3.4 Backend Implementation

— **Spring Boot REST API :** Handles CRUD operations for cars, agencies, bookings, and blogs.

— **Security :**

  — Keycloak handles login and registration for admins and agencies.

  — AuthGuard in Angular protects routes based on roles.

  — Only authorized users can access admin or agency dashboards.

— **Booking Workflow :**

  — Customer requests a car rental ; the system checks car availability via `getUnavailableDates()`.

  — Request is sent to the responsible agency.

  — Agency accepts/declines the request ; email notifications are sent accordingly.

  — Accepted requests generate PDF contracts with a QR code sent to the customer.

— **Real-time Communication :** Admins and agencies communicate via instant messaging.

— **Chatbot :** Integrated via external Flask API, accessible to visitors and customers for queries.

3.5 Frontend Implementation

— Angular components handle car listings, rental requests, blogs, notifications, and chat.

— Reactive forms validate inputs for registration, login, booking, and comments.

— Guards ensure route protection according to user roles.

— Email and PDF confirmations are triggered using backend API services.

3.6 Deployment Strategy

— **CI/CD :** GitLab pipelines automate build and deployment.

— **Docker :** Separate containers for Angular frontend, Spring Boot backend, MySQL, and Flask chatbot.

— **Kubernetes :** Minikube manages deployments with YAML files such as `mysql-deployment.yml`, `frontend-deployment.yml`, `backend-deployment.yml`, `ai-deployment.yml`.

3.7 Development Methodology

— Scrum methodology with iterative sprints and backlog management.

— Team roles : Scrum Master, Developers, Product Owner.

— Regular sprint reviews and retrospectives ensure continuous improvement.

3.8 Conclusion

This chapter demonstrates the technical setup, database design, backend and frontend implementation, deployment strategy, and development methodology for the Car Rental Platform. It ensures secure, real-time, and scalable service delivery to admins, agencies, customers, and visitors.

Chapitre 4

Implementation and Results

Plan

4.1 Introduction

This chapter presents the practical implementation of the Car Rental Platform. It details how the system architecture, design decisions, and core functionalities have been translated into a working product. The chapter is organized by module, describing role-specific components, backend logic, frontend interaction, security integration, third-party service connections, and test scenarios. Screenshots and results are included to illustrate the platform's functionalities.

4.2 System Modules Implementation

The application is organized into modular components corresponding to specific functionalities, user roles, and workflows. Each module is independently testable, scalable, and maintained in alignment with the microservice architecture.

4.2.1 Authentication and Authorization Module (Keycloak)

— Role-based access control is implemented using Keycloak.

— Users are authenticated via OAuth2 and issued JWT tokens.

— Angular AuthGuard protects Admin, Agency, and Customer dashboards.

— Spring Boot backend routes are secured with role-based annotations (e.g., `@PreAuthorize`, `@RolesAllowed`).

4.2.2 Admin Module

— Add, update, and delete agencies.

— Manage customers and blogs (CRUD operations).

— View and manage all bookings.

— Communicate with agencies via real-time chat.

4.2.3 Agency Module

— Add, edit, and delete cars.

— Upload photos for existing cars.

— Manage availability and booking requests.

— Accept or decline rental requests, triggering email and notification.

— Communicate with admin via real-time chat interface.

4.2.4 Customer Module

— Browse available cars and blogs.

— Register and log in with Keycloak (or browse as a visitor).

— Submit rental requests for available cars.

— Receive notifications and emails upon status changes.

— View rental history and delete unconfirmed or declined requests.

— Comment on car listings and blogs.

— Receive PDF contract with QR code after payment confirmation.

4.2.5 Chatbot Module (FastAPI + ChatGPT)

— FastAPI backend receives questions from the frontend.

— OpenAI's ChatGPT API generates responses.

— Replies are displayed to customers and visitors.

— Supports multilingual queries (French and English).

4.2.6 Online Payment Module

— Upon booking acceptance, a payment link is sent to the user's email.

— Payment is processed and validated through the Spring Boot backend.

— A PDF with QR code is generated and sent after payment confirmation.

— Booking status updated to "confirmed".

4.2.7 Notification and Email System

— Customers receive emails for new cars, booking decisions, or updates.

— Followers (visitors who submit emails) receive notifications for new car listings.

— Notifications are stored in the database and displayed in-app.

4.3 Database Implementation

The platform uses a MySQL relational database. Entities are mapped via JPA annotations and structured around key objects :

— **Agence Table :** Stores agency profiles and login credentials.

— **Booking Table :** Stores rental requests, dates, customer info, price, and car details.

— **Blog Table :** Contains articles, authors, multimedia content.

— **Voiture Table :** Stores car details (name, type, price, photos).

— **User Table :** Stores customer information and login references.

— **Notification Table :** Stores in-app messages and alert statuses.

4.4 Screenshots of the Application

Screenshots illustrate the platform's functionalities for different user roles :

— Admin dashboard with agency management.

— Agency dashboard with car CRUD operations.

— Customer booking page and rental history view.

— Real-time chat interface between admin and agencies.

— Generated PDF contract with QR code.

— Chatbot interface integrated on the homepage.

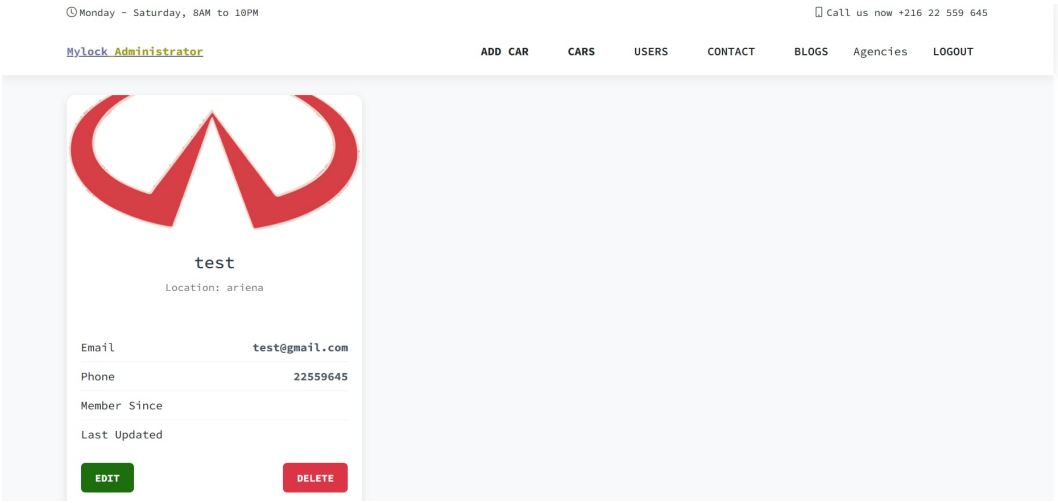

Figure 4.1 : Admin Dashboard for Managing Agencies

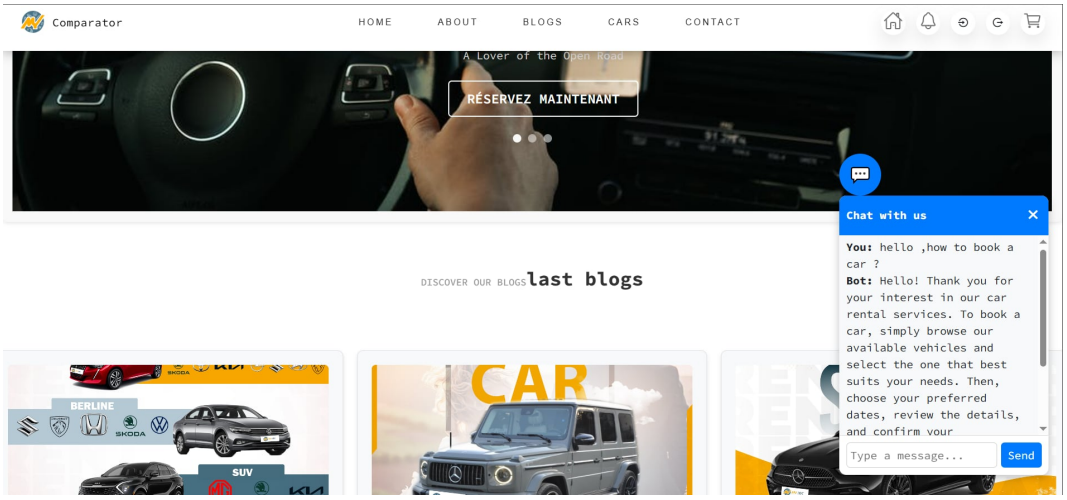

Figure 4.2 : Chatbot Integration on Homepage

4.5 Testing and Validation

The system has been validated through manual and automated testing.

4.5.1 Unit Testing

— Spring Boot unit tests for services and controllers.

— REST API endpoints tested with Postman.

— Angular unit tests using Jasmine and Karma.

4.5.2 Integration Testing

— End-to-end testing of booking flow from customer request to agency validation.

— Payment processing, PDF generation, and email dispatch tested with mock gateways.

— Real-time chat functionality tested between admin and agencies.

4.5.3 CI/CD Pipeline Validation

— GitLab CI pipelines test build processes, run unit tests, and deploy to Kubernetes.

— Docker containers tested locally and in staging environments.

— Rollback and hotfix procedures validated.

4.6 Conclusion

The implementation phase successfully transformed the design into a robust, functional web platform. All planned modules—from authentication and booking workflows to real-time messaging, automated email notifications, chatbot support, and online payments—have been developed, tested, and deployed. The system is stable, scalable, and provides a seamless digital experience for all user roles.

Chapitre 5

Conclusion and Future Work

Plan

5.1 Summary of the Project

This project developed a comprehensive Car Rental Platform that integrates multiple user roles, real-time communication, online payment processing, automated notifications, and AI-powered assistance. The platform enables :

- Admins to manage agencies, customers, blogs, and oversee bookings.
- Agencies to manage cars, accept or decline rental requests, and communicate with admins.
- Customers to browse cars, make rental requests, view history, comment, and receive automated emails and PDF confirmations.
- Visitors and customers to interact with a chatbot powered by ChatGPT for assistance.

The platform is developed using Angular for the frontend, Spring Boot for the backend, MySQL for the database, Keycloak for authentication, and FastAPI for the chatbot service. The system is containerized with Docker and deployed on Kubernetes using a CI/CD pipeline.

5.2 Key Achievements

- Successful implementation of role-based access control and secure authentication using Keycloak.
- Development of a real-time chat system for admin-agency communication.
- Automated booking workflow with email notifications and PDF contract generation.
- Integration of a multilingual AI chatbot for user assistance.
- Deployment of a scalable microservices architecture using Docker and Kubernetes, including :
  - Separate containers for frontend, backend, database, and AI services.
  - YAML deployment files for Minikube (`frontend-deployment.yml`, `backend-deployment.yml`, `mysql-deployment.yml`, `ai-deployment.yml`).
  - CI/CD pipeline using GitLab for automated builds, testing, and deployment.
- Implementation of unit, integration, and end-to-end testing to ensure system reliability.

5.3 Limitations

- Online payment module relies on mock gateways in the current implementation ; full integration with external payment providers is not yet complete.
- Chatbot responses may depend on external API latency and internet connectivity.
- Mobile responsiveness and offline capabilities can be further enhanced.
- Some modules could be optimized for performance under high concurrent user loads.

5.4 Future Work

— Integrate real payment gateways for live transactions and invoicing.

— Enhance the chatbot with advanced NLP capabilities and additional languages.

— Implement push notifications for mobile devices.

— Add analytics dashboards for admins and agencies to track usage, bookings, and revenue.

— Introduce recommendation systems to suggest cars to customers based on preferences and history.

— Optimize microservices performance and scalability for large-scale deployment.

— Improve deployment automation and monitoring using Kubernetes tools such as Helm, Prometheus, and Grafana.

5.5 Final Remarks

The Car Rental Platform provides a robust, secure, and user-friendly system that automates key rental workflows and integrates modern technologies such as AI and microservices. The deployment process demonstrates the ability to manage containerized applications, orchestrate them with Kubernetes, and use CI/CD pipelines for seamless updates. This platform serves as a solid foundation for future enhancements and real-world deployment in the car rental industry.

5.5.1 User Testimonials

To complement quantitative testing results, we collected feedback from real users who interacted with the platform. Selected testimonials include :

- **Bassem Ajengui (CEO & Founder) :** "Super application, très intuitive et facile à utiliser. Je recommande vivement !"
- **Assyl Kria (Designer) :** "Franchement, rien à dire ! Les voitures sont top et le service hyper pro."
- **Bayrem Boussaidi (Store Owner) :** "J'ai adoré la facilité de réservation et la qualité des véhicules proposés."
- **Aymen Arfaoui (Freelancer) :** "Une des meilleures agences de location de voitures, je reviendrai sans hésiter."
- **Mouhib Touati (Entrepreneur) :** "Je suis très satisfait du service, les prix sont abordables et les voitures impeccables."

These testimonials demonstrate positive reception from multiple user roles, confirming the platform's usability, clarity, and reliability.

Conclusion générale

Ce projet avait pour objectif de concevoir, développer et déployer une plateforme web complète de location de voitures, adaptée aux besoins de différents utilisateurs : administrateurs, agences, clients et visiteurs. En suivant une méthodologie Agile Scrum et en s'appuyant sur des outils DevOps modernes, nous avons réussi à mener à bien toutes les phases, depuis l'analyse des besoins jusqu'au déploiement sur un cluster Kubernetes.

La plateforme intègre avec succès plusieurs fonctionnalités clés :

— Authentification sécurisée multi-rôles et contrôle d'accès via Keycloak.

— Gestion des véhicules par les administrateurs et agences, incluant la gestion des photos.

— Processus de réservation fluide avec vérification en temps réel de la disponibilité et validation par l'agence.

— Notifications automatiques par email et génération de contrats PDF avec QR code intégré.

— Système de chat en temps réel facilitant la communication entre agences et administrateurs.

— Chatbot intelligent offrant un support 24h/24 via une intégration FastAPI et OpenAI.

— Pipelines CI/CD robustes utilisant GitLab, la containerisation Docker et un déploiement scalable via Kubernetes.

Ce projet a permis de combiner avec succès architecture RESTful, modularité microservices et outils d'intelligence artificielle pour offrir une solution complète, scalable et maintenable. Il ouvre la voie à de nombreuses améliorations futures tant techniques que fonctionnelles.

Ainsi, ce travail illustre parfaitement comment les technologies web modernes et les principes du cloud peuvent transformer un processus traditionnel de location en une expérience numérique avancée, apportant valeur ajoutée aux utilisateurs et aux gestionnaires du service.

Bibliographie

[B1] Pierre Pezziardi, Référentiel des Pratiques Agiles, édition ebook.2013

Résumé

FR

Mots clés : KW1, KW2

Abstract

EN

Keywords : KW1, KW2