People's Democratic Republic of Algeria
Ministery of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Computer Science
Department of Computer Science

DISSERTATION

Presented in fulfillment of the requirements of obtaining the degree

Master in Computer Science

Specialty: Networking & Multimedia

THEME

Design and Implementation of an Innovative Digital Platform for Vehicle Breakdown Assistance

Presented by:

Oussama RAHIM

Mohammed salah eddine SAIDANI

Publicly defended on: 09/07/2025

In front of the jury composed of:

President: Dr. Meriem LAIFA

Examiner 1: Dr. Boubakeur MOUSSAOUI

Examiner 2: Dr. Abdallah BENGUEDDOUDJ

Supervisor: Dr. Nadjib BENAOUDA

2024/2025

Dedication

This work is a testament to the unwavering support and inspiration provided by those who have walked alongside me on this academic and personal journey.

To my beloved father, whose strength and sacrifices have been a guiding light, and to my cherished mother, whose boundless love and nurturing spirit have given me the courage to pursue my dreams.

To my dear sister and brothers, your encouragement, laughter, and shared moments have been my anchor through every challenge.

To my best friend, Mohammed SAIDANI , whose steadfast camaraderie, endless conversations, and belief in me have been a source of joy and motivation.

To my esteemed supervisor, Dr. Nadjib BENAOUDA, whose profound wisdom, patient guidance, and unwavering dedication to academic excellence have shaped this project into what it is today.

To all the professors at Mohamed El Bachir El Ibrahimi University, whose knowledge, mentorship, and commitment to fostering innovation have inspired me to push the boundaries of possibility.

To my dear friends, who have offered support, shared in my triumphs, and stood by me through every hurdle, your presence has made this journey meaningful.

This project, aimed at revolutionizing roadside assistance in Algeria, is dedicated to all of you, whose collective influence has fueled my passion for creating solutions that enhance connectivity and progress in our community.

Oussama RAHIM

Dedication

In the tapestry of life, where threads of love, guidance, and inspiration are woven into the fabric of my being, I dedicate this masters thesis to those who have colored my journey with hues of wisdom and warmth.

To my father, the steadfast mountain whose peaks have guided my ambitions, your unwavering strength has been my shelter through every storm. To my mother, the gentle river that carves paths through stone, your nurturing flow has shaped my dreams with grace and resilience.

To my brother, the steadfast beacon in my sky, your unwavering light has guided me through challenges, illuminating my path with courage and camaraderie. To my Elder sister, the vibrant sunrise of my days and my esteemed professor and former supervisor, your wisdom has not only painted my world with hope and joy but also shaped my academic journey with unparalleled insight and mentorship. To my second sister, the serene moonlight of my nights, your quiet strength and grace have soothed my heart through every trial.

To my best friend and partner, Oussama RAHIM , the compass of my heart,your unyielding support has charted the course of this endeavor, turning dreams into reality with every shared step.

To my esteemed supervisor, Dr. Nadjib BENAOUDA, the master cartographer of knowledge, your wisdom has drawn the map for this academic voyage, illuminating paths I could not have navigated alone.

To all the professors at Mohamed El Bachir El Ibrahimi University, the architects of intellect, your collective guidance has built the foundation upon which this work stands.

To my dear friends, who have celebrated my successes, and remained steadfast through every challenge, your companionship has given this journey profound significance.

This endeavor, designed to transform roadside assistance in Algeria, is devoted to each of you, whose

combined inspiration has ignited my drive to develop solutions that foster connection and advancement within our society.

Mohammed salah eddine SAIDANI

Acknowledgment

This work is a testament to the unwavering support and inspiration provided by those who have walked alongside me on this academic and personal journey.

We express our deepest gratitude to all those who have supported us throughout the journey of completing this dissertation. First and foremost, we sincerely thank our supervisor for their invaluable guidance, expertise, and encouragement, which were instrumental in shaping this project. Their insightful feedback and dedication inspired us to push the boundaries of our research and development.

We are also grateful to the faculty members of the Department of Computer Science at Mohamed ElBachir ElIbrahimi University for their academic support and for fostering an environment conducive to learning and innovation. Our appreciation extends to the jury members for their time, constructive feedback, and expertise in evaluating our work.

Wewouldlike to acknowledge our peers and colleagues who provided technical assistance, shared ideas, and offered encouragement during the development of the digital platform. Their collaboration and diverse perspectives enriched our project and made the process more rewarding.

On a personal note, we extend our heartfelt thanks to our families and friends for their unwavering support, patience, and belief in us. Their encouragement provided the strength and motivation to overcome challenges and complete this academic milestone.

Finally, we dedicate this work to the people of Algeria, whose need for efficient and reliable roadside assistance inspired this project. We hope our platform contributes to enhancing connectivity and support within our community.

Oussama RAHIM & Mohammed salah eddine SAIDANI

Abstract

This project introduces a digital platform to transform vehicle breakdown assistance in Algeria, addressing inefficiencies in traditional roadside services. It comprises three mobile applications for stranded motorists, autonomous repairmen, and workshop repairmen, and two web interfaces for workshop and platform administrators, built using Flutter, Next.js, Node.js with Express.js, and Firebase Realtime Database. The platform ensures scalability and real-time synchronization, offering features like GPS-based technician matching, secure user registration, a cash-based payment system, and performance analytics. Targeting motorists, repairmen, and administrators, it streamlines service requests and operational management. Analysis of existing solutions like DZ Dépannage revealed gaps in real-time tracking and payment systems, which this platform addresses with a user-centric design. The platform was developed using the Scrum framework across six sprints, utilizing a NoSQL database for flexible data management and efficient geolocation-based service provision.

Keywords: Roadside Assistance in Algeria, Scrum framework, Mobile development, Web development, Firebase Realtime Database.

Résumé

Ce projet présente une plateforme numérique qui a pour objectif de transformer l'assistance en cas de panne de véhicule en Algérie, en remédiant aux lacunes des services de dépannage classiques. Elle se compose de trois applications mobiles conçues pour les conducteurs en situation de panne, les dépanneurs indépendants et les techniciens en atelier, accompagnées de deux interfaces web dédiées aux responsables d'ateliers et aux administrateurs de la plateforme. La solution a été conçue en utilisant les technologies Flutter, Next.js, Node.js avec Express.js, et la base de données en temps réel de Firebase. Elle garantit une montée en charge efficace et une synchronisation en temps réel, tout en proposant des fonctionnalités telles que la mise en relation des techniciens basée sur la géolocalisation, une inscription sécurisée des utilisateurs, un système de paiement en espèces, et des outils d'analyse de la performance. En ciblant les conducteurs, les dépanneurs et les gestionnaires, la plateforme vise à simplifier les demandes d'assistance et la gestion opérationnelle. L'examen des solutions déjà disponibles telles que DZ Dépannage a révélé des insuffisances concernant le suivi en temps réel et les systèmes de paiement, lacunes qui sont comblées par cette plateforme grâce à une approche centrée sur l'utilisateur. La plateforme a été conçue en suivant la méthodologie Scrum, avec un total de six sprints. Elle repose sur l'utilisation d'une base de données NoSQL afin d'assurer une gestion souple des données et une prestation efficace de services basés sur la géolocalisation.

Mots-clés: Assistance routière en Algérie, cadre Scrum, développement mobile, développement web, Firebase Realtime Database.

ملخص

يقدّم هذا المشروع منصة رقمية لتحسين خدمات المساعدة عند تعطل المركبات في الجزائر، وذلك من خلال معالجة أوجه القصور في الخدمات التقليدية على الطرقات. تتكوّن المنصة من ثلاث تطبيقات للهاتف المحمول مخصصة للسائقين المتعطلين، والميكانيكيين المستقلين، وورشات الإصلاح، بالإضافة إلى واجهتين ويب لإدارة الورشات والمنصة، وقد تم تطويرها باستخدام Flutter ، Next.js ، Node.js مع Express.js وقاعدة بيانات Firebase Realtime Database. توفّر المنصة قابلية للتوسع ومزامنة فورية، وتضم ميزات مثل مطابقة الفنيين باستخدام نظام تحديد المواقع GPS، تسجيل المستخدمين بشكل آمن، نظام دفع نقدي، وتحليلات أداء. تستهدف المنصة السائقين، الميكانيكيين، والمشرفين، وتهدف إلى تبسيط طلبات الخدمة وإدارة العمليات. أظهر تحليل للحلول الحالية مثل "DZ Dépannage" وجود ثغرات في التتبع اللحظي وأنظمة الدفع، وهي نقاط ضعف تتجاوزها هذه المنصة بفضل تصميم يركز على تجربة المستخدم. تم تطوير المنصة باستخدام إطار عمل Scrum على مدار ستة سباقات تطوير (Sprints)، مع الاستعانة بقاعدة بيانات NoSQL لتوفير إدارة بيانات مرنة وخدمة فعالة تعتمد على الموقع الجغرافي.

الكلمات المفتاحية: المساعدة على الطريق في الجزائر، إطار عمل Scrum، تطوير تطبيقات الهاتف المحمول، تطوير الويب، Firebase Realtime database.

Table of Contents

List of Figures

List of Tables

Chapter 1

General Introduction

1.1 Background

Today, technology plays a pivotal position in almost all aspects of our life. Digitization has emerged as a crucial instrument for development, revolutionizing key sectors including education, health, business, and particularly tourism. The rapid growth of web and mobile apps has transformed how services are delivered, providing innovative solutions to everyday challenges. Faster access to information, service automation, and streamlined communication are now fundamental requirements in contemporary interactions.

Amid the global move to digital technology, Algeria is progressively initiating its digital transformation. An increasing number of sectors, both public and commercial, are adopting digital technologies to enhance service quality and more effectively fulfill citizens' expectations. Within these developing sectors, transportation and roadside assistance present a significant opportunity to augment safety, deliver expedited help to drivers, and elevate overall mobility services.

The increasing demand for efficiency and dependability in roadside services highlights the need for digital solutions. It is in this direction that our work is oriented, seeking to improve roadside assistance in Algeria with an intuitive and interconnected platform.

1.2 Problem statement

Mechanical issues, collisions, tire punctures, fuel deficiencies or mistakes, and even the loss or theft of vehicle keys are just a few examples of incidents that can leave a vehicle immobile and its driver in need of urgent assistance. When such situations occur, it's essential to get prompt roadside assistance. However, in Algeria, the current roadside assistance system faces several challenges: slow response times, limited service availability, and a lack of clear communication between stranded drivers and repair teams. These issues raise important questions: What is the appropriate response ? What is the expense associated with roadside assistance or towing ? and what compensation can be anticipated from your insurance policy?

Besides these general challenges, the main operational issues faced by roadside assistance can be outlined as follows:

- The stranded motorist is often unaware of how to locate a competent, available, and reliable technician nearby, nor of the expected cost of assistance.
- Despite the availability of numerous roadside technicians, finding one knowledgeable about the specific issue and able to respond quickly remains challenging.
- As a result, considerable time may be wasted attempting to locate a suitable roadside technician.
- The exact location of the stranded driver may not be known to the roadside technician, leading to further delays.
- The technician is usually not informed in advance about the specific nature of the malfunction (e.g., fuel depletion, battery failure, tire deflation), which affects their preparedness.

All these problems stem from conventional search methodologies, which highlight the difficulty of finding a roadside expert to aid a stranded driver, especially in emergencies.

On the other hand, the ubiquity of the Internet, the advent of the web, and the widespread use of smartphones have enabled the utilization of digital solutions through mobile or web applications to intelligently assist stranded motorists and roadside technicians, ensuring efficient, rapid, and straightforward roadside assistance.

Consequently, our efforts are directed on presenting digital solutions that optimize the roadside assistance process by offering all necessary functionalities for both drivers and service providers, ensuring a swift and efficient response.

1.3 Objective and methodology

The objective of this project is to design and implement an innovative digital platform for vehicle breakdown assistance, overcoming the shortcomings of existing solutions. The platform provides a multi-platform experience through three mobile applications for stranded motorists, autonomous repairmen, and workshop repairmen, and two web interfaces for workshop and platform administrators, enabling motorists to request assistance, track technicians in real-time, and access transparent pricing, while supporting repairmen and administrators in managing services efficiently. The development process adopted an Agile methodology with the Scrum framework, involving requirements gathering via stakeholder analysis, iterative development across six sprints, and continuous feedback to align with user needs. The technology stack includes Next.js for the web client, Flutter for the mobile app, Node.js and Express.js for the backend, and Firebase for scalable data storage, integrated with third-party APIs for location tracking and validation of email and phone number.

1.4 Report structure

This report is structured to provide a comprehensive overview of the project. While Chapter 1 serves as a general introduction, the remaining chapters are organized as follows:

- Chapter 2 evaluates existing roadside assistance solutions, defines project goals, identifies targeted users, and outlines functional and non-functional requirements.
- Chapter 3 details the Agile and Scrum methodology, its application in our development process, use case diagrams along with system sequence diagrams for selected use cases, and the Firebase database design.
- Chapter 4 describes the implementation, including the system architecture, technology stack, API routes, user flow diagrams, and key application interfaces.
- Chapter 5 offers the general conclusion, summarizing contributions, limitations, and fu-

ture perspectives.

This organization guides the reader through the projects conceptualization, design, and implementation.

Chapter 2

Analysis of the existing solutions and requirements specification

2.1 Introduction

In this chapter, we outline the overall framework for our project, whose objective, as previously articulated, is to design and implement an innovative digital platform for vehicle breakdown assistance. Initially, we will dive straight into the current situation, aiming to assess the landscape in Algeria, with particular emphasis on existing solutions and market gaps. We will then detail the specific goals we aim to address with our solution. The subsequent phase entails delineating the targeted users. The identification of functional and non-functional needs concludes this chapter.

2.2 Existing solutions

This section outlines currently available digital solutions available to Algerian inhabitants for assistance after a vehicular breakdown. Our study and exploration revealed two main mobile applications, which will be discussed in the subsequent two sections.

Besides these two solutions, other conventional methods can be highlighted. For example, locating technicians or workshops through Google Maps, or using the emergency services provided by insurance companies. However, these solutions will not be detailed here, as the focus

is on dedicated platform-based solutions for roadside breakdown assistance.

2.2.1 Dépannage dz

Presentation

DZ Dépannage[1] is an Algerian service that enables motorists to travel with confidence. The application offers a compilation of experts, tow truck operators, and tire specialists tailored to the user's requirements. This program enables you to locate roadside help based on your wilaya and municipality, regardless of your location. 'DZ Dépannage' was created by the Fikra Plus+ team and introduced in 2016.

Shortcomings

DZ Dépannage lacks full assistance functions, highlighting a significant market deficiency. In contrast to our suggested application, which features GPS monitoring, real-time service updates, in-app communication, and safety advise, DZ Dépannage offers a more rudimentary and restricted functionality. The lack of advanced support choices renders DZ Dépannage less effective for those needing prompt, well-coordinated actions during automobile crises. Our software seeks to address these deficiencies by launching a comprehensive platform that spans both mobile and web, providing motorists with a more versatile, accessible, and responsive option for breakdown help.

2.2.2 Depani Dz

Presentation

Dépani[2] is an intuitive mobile application that links motorists with nearby roadside assistance services and auto maintenance establishments around Algeria. The software facilitates the swift and straightforward identification of vehicle repair services, enabling users to seek nearby professionals and workshops tailored to their requirements, access critical safety protocols, and display pertinent emergency contact information. Service providers, such as tow truck operators and mechanics, can effortlessly register on the platform to connect with local

[1] https://play.google.com/store/apps/details?id=com.depannage_dz&pcampaignid=web_share
[2] https://play.google.com/store/apps/details?id=com.zo.depani&pcampaignid=web_share

users seeking assistance. Dépani seeks to establish a convenient and efficient experience for all automotive assistance needs.

Shortcomings

While offering fundamental vehicle breakdown help, there exist numerous significant market deficiencies that hinder its service efficacy and consumer convenience. Dépani mostly functions through call-based support, lacking crucial current features like as GPS mapping, which would facilitate precise position sharing and enable technicians to route effortlessly to consumers. The lack of map-based services leads to delayed reaction times and diminished accuracy in service coordination. Furthermore, Dépani does not offer transparent pricing, resulting in ambiguity for users who favor understanding service charges prior to engaging support. Dépani's platform, without these advanced elements, is fundamentally simple, presenting a chance for a superior solution that incorporates these absent functionalities, hence providing a more transparent, user-friendly, and efficient experience.

2.2.3 Synthesis

Our targeted application rectifies the significant deficiencies of Dépani and DZ Dépannage by providing a contemporary, all-encompassing solution for vehicle breakdown support. Our application will utilize GPS mapping, facilitating stranded motorists to immediately disclose their exact location and enabling technicians to locate them effectively. This real-time monitoring decreases wait times and improves service precision. Furthermore, our application will present service rates transparently, instilling consumers with assurance in their decisions. Our app will offer an intuitive layout and comprehensive functionality, featuring secure in-app payments, real-time updates, and safety instructions, so delivering a more responsive, dependable, and user-focused experience. This revolutionary approach not only fulfills but also exceeds the fundamental requirements of motorists, distinguishing our platform as the premier option in vehicle breakdown support.

2.3 Project Goals

All individuals are susceptible to mechanical failures when traversing extensive distances in their vehicles. A motorist may encounter a breakdown or technical fault with their vehicle,

often necessitating the help of a mechanic for roadside support.

A digital solution for roadside assistance help streamlines and improves the process by providing an efficient, rapid, and tailored solution for locating and arranging auto repair services.

Below, we enumerate our set short- and long-term objectives that we want to address throughout our project.

2.3.1 Short-Term Goals

In this project, we aims to provide an IT solution assistance website and mobile application (for Android and iOS) to primarily address the following short-term objectives:

- Facilitate digital support through a broader, more transparent, and intuitive interface, enhancing the efficiency, simplicity, and speed of roadside help.
- Enable users to locate and identify a qualified, available roadside technician.
- Assist stranded motorists in requesting a technician at any time and from any location.
- Assist the technician in accurately identifying the location of the stranded driver and understanding the nature of the problem beforehand.
- Facilitate effective communication and cooperation between the stranded motorist and the technician.
- Provide a complete solution for service companies to handle stranded motorists' requests by allocating the appropriate technician to each task according to their availability and expertise.
- Optimize the roadside assistance process to conserve time and financial resources.
- Enhance the comprehensive roadside help experience.
- Allow drivers to evaluate the performance of automotive specialists based on job quality and to access servicing costs.
- Offer package deals for regular roadside assistance services, helping users save money on frequent breakdowns or issues.

2.3.2 Long-Term Goals

For the long-term goals, our aim is to address the following:

- Provide a 24/7 chat service or helpline where users can get advice or guidance on what steps to take while waiting for assistance
- Include a feature that prioritizes technicians using electric or eco-friendly vehicles to reduce the environmental impact.
- Make partnership with big workshops (like GME-perfomance, GE-performance.. ect) that provides opportunities for both workshop and rode-riders.

2.4 Targeted users

This section presents the identified users who intend to utilize our platform. We delineate each individual's role and responsibilities, along with the various features that have to be offered by the platform to assist users in fulfilling their tasks. Four primary user categories are identified: stranded motorists, automotive technicians (autonomous and workshop repairmen), workshops or service companies, and administrators.

2.4.1 Stranded Motorists

Role: Motorists are the primary customers, using the platform on a mobile app to request and receive roadside assistance when needed.

Responsibilities:

- Register or log into the platform.
- Get a nearby help by available technician from the list of options (based on availability, proximity).
- Provide details of the vehicle problem (e.g., flat tire, dead battery, accident).
- Share the exact location (via GPS) to help the technician find them easily.
- Track the technician's arrival in real-time.

- Pay for the service through the integrated payment system.
- Rate and review the technician's service post-assistance.

Features for Motorists:

- User Registration, Authentication and Profile Management.
- Simple request submission with problem categorization.
- GPS tracking and real-time technician location updates.
- Secure payment gateway integrated within the app.
- Service history to view previous requests and resolutions.
- Feedback and rating system for technicians.

2.4.2 Autonomous Repairman

Role: Provides necessary roadside assistance, including diagnosing and resolving the vehicle's issue (mechanical repairs, towing, or other support). They utilize a customized mobile application to connect with stranded motorists. It provides necessary roadside assistance independently without being affiliated with a workshop.

Responsibilities:

- Request registration via a form.
- Login to the application.
- Receive service requests based on their location and availability.
- View the exact location and problem description provided by the motorist.
- Communicate directly with the motorist through the platform to confirm details or arrival time.
- Update the status of the intervention (on the way, completed, etc.).

- Input details of the service performed (e.g., type of repair) and confirm payment if necessary.

Features :

- Technician registration, authentication and profile management.
- Service tracking and management interface (current and past interventions).
- Real-time navigation and location sharing.
- Payment processing or confirmation interface.
- Communication tools (chat/call with the motorist).

2.4.3 Workshop Repairman

Role: Performs roadside repairs or towing under the supervision of a workshop.

Responsibilities: He has mostly the same responsibilities and features as the Autonomous Repairman, the deference's are :

- Register and log in through a workshop account.
- Accept and complete service requests assigned by the workshop.
- Update the service status for tracking purposes.
- Coordinate with workshop administrators and communicate with customers.
- Ensure timely and efficient roadside assistance.

Features :

- Works under a structured schedule set by the workshop.
- Follows workshop pricing and service guidelines.
- Send the billing to workshop.

2.4.4 Workshop Administrator

Role: Workshops and service companies correspond to enterprises that provide roadside assistance services. They are in charge of a team of technicians, monitor all service requests, and make sure that the work is done quickly and well via a web platform (website).

Responsibilities:

- Request registration.
- Login to website.
- Assign technicians to incoming service requests based on availability and expertise.
- Track technician locations and status for efficient scheduling and dispatch.
- Review service performance, gather customer feedback, and implement improvements.
- Ensure that response times and service quality align with company standards.
- Manage billing for completed services, including confirming payment processing.

Features for Workshops:

- Registration, authentication and profile management.
- Centralized interface to monitor and assign technicians to incoming service requests.
- Real-time overview of technician locations, status, and availability.
- View and manage all active, pending, and completed service requests.
- Access to request history, including details of each intervention, for performance tracking and analysis.
- Generate reports on technician performance, response times, and service quality.
- Access billing history, manage refunds if necessary, and ensure financial transparency.
- Provide chat or call features to facilitate communication between workshop managers and technicians.

- Offer guidance and support as needed for more complex service issues.

2.4.5 Administrator

Role: Oversees the operation and functionality of the platform, ensuring smooth coordination between motorists and technicians, via a web platform (website).

Responsibilities:

- Login to platform.
- Manage and approve technician registrations.
- Monitor platform activity (requests, services in progress, and completed tasks).
- Handle any disputes or complaints from motorists or technicians.
- Update the platform's database of service providers (adding new ones, removing inactive providers, etc.).
- Ensure timely updates to the app's functionality, manage user data, and ensure payment security.
- Generate reports on platform usage, financial performance, and user feedback.

Features for Administrators:

- Authentication and profile management.
- Dashboard to view real-time platform activity and statistics.
- Technician management (profile verification, activation/deactivation).
- Analytics and reporting tools.
- Customer support interface for dispute resolution.
- Maintenance tools to update pricing, service areas, and other variables.

2.5 Requirements Specification

This section outlines the functional and non-functional requirements that our platform must satisfy. Functional requirements refer to the operations and features that must be implemented to fulfill user needs. Non-functional criteria specify how the system should behave or perform, encompassing essential quality features such as security, scalability, dependability, usability, and overall system performance [1].

2.5.1 Functional Needs

- **User Registration and Authentication :** Allow users (motorists, technicians, and admins) to register, log in, and manage their profiles securely.
- **Service Request Creation :** Enable motorists to request breakdown assistance, specifying their issue (e.g., battery problem, flat tire, towing needs) and location.
- **GPS Mapping and Real-Time Location Tracking :** Use GPS to display the motorist's location and enable technicians to navigate directly to the site.
- **Technician Matching and Availability :** Match motorists with nearby, available technicians and show options based on proximity, ratings, or cost.
- **In-App Communication :** Allow motorists and technicians to communicate directly (via chat or call) to confirm details.
- **Service Status Updates :** Track service progress with updates (e.g., technician en route, arrived, completed) visible to both the motorist and technician.
- **Payment System :** Provide a secure, integrated payment gateway to finalize payments.
- **Service Rating and Review System :** Enable users to rate and review technicians after service completion, building trust and credibility on the platform.
- **Admin Dashboard :** Provide administrators with tools to monitor platform activity, manage users, and oversee service operations.

2.5.2 Non-Functional Needs

- **Performance and Scalability :** Ensure the website handles high traffic and scales as more users join, maintaining quick response times.
- **Security and Data Protection :** Implement strong security measures, including data encryption, secure payment processing, and GDPR compliance for data privacy.
- **Reliability and Availability :** Guarantee high availability (e.g., 99.9% uptime), with a robust server infrastructure to ensure the website is accessible at all times.
- **User Experience (UX) and Usability :** Design an intuitive, responsive interface that provides a seamless experience across devices, with easy navigation and clear instructions.
- **Compatibility :** Ensure that the mobile applications are compatible with the most widely used operating systems (iOS and Android) and support a range of system versions to maximize accessibility and usability for different user devices. Additionally, ensure the websites are compatible across all major browsers and supports different screen resolutions, including mobile and desktop versions.
- **Maintainability :** Structure the codebase and documentation to support easy maintenance and future updates, allowing for quick bug fixes and feature enhancements.
- **Localization and Language Support :** Provide multi-language support to cater to a diverse user base, adapting content and interface to the local language and culture.
- **Audit Logging and Tracking :** Track all user interactions and service history for troubleshooting, compliance, and performance assessment purposes.

These requirements will ensure the platform is functional, secure, reliable, and user-friendly, meeting both the operational and quality expectations of its users.

2.6 Conclusion

This chapter delineated the fundamental prerequisites for our vehicle breakdown support platform, emphasizing the critical functional and non-functional criteria. We included essential

features including real-time GPS tracking, secure user administration, service request processing, and a dependable payment mechanism to guarantee a user-friendly and efficient experience for all stakeholders. Furthermore, we recognized non-functional requirements, encompassing performance, security, and compatibility criteria, to enhance the platform's quality and scalability. Collectively, these prerequisites establish a robust framework for the design and execution of the platform.

In the subsequent chapter, we will mainly present the adopted methodology for implementing our platform, namely the Scrum Agile framework, along with its practical application throughout the development process.

Chapter 3

Methodology and Design

3.1 Introduction

This chapter outlines the methodology we employed to implement the distinctive features of our platform. The Scrum process was chosen for this purpose because of its widespread adoption and appropriateness for such development projects.

This chapter starts with a succinct review of agile approaches, emphasizing Scrum specifically. The subsequent section presents several use case diagrams that depict the functionalities of each actor inside the platform, accompanied by system sequence diagrams for selected use cases. Based on these diagrams, we then describe how Scrum was applied throughout the development process. The chapter ends with the design of our database, encompassing both the conceptual data model, which delineates all needed data for implementing the stated features and the logical data model, which translates this conceptual model to the particulars of a NoSQL database.

3.2 Agile methodologies overview

3.2.1 Agile definition

Agile project management methodology, first introduced in the late 1980s, is characterized by a non-linear approach. Instead of being completed in the *Planning* step, requirements and solutions evolve throughout the process through collaboration between self-organizing, cross-

functional teams.

Execution of project work is done continuously throughout the life of the project. Agile methods involve adaptive planning, evolutionary development, early delivery, and continuous improvement. Agile encourages rapid and flexible response to change.

This method is increasingly popular and is often used to manage the process of creating software. It focuses on teamwork and finishing one feature of the software at a time (fully) before moving onto the next [2].

The Agile philosophy is best captured in the Agile Manifesto [3], which identifies the following values:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile is a descriptor of approaches that align with the values of the Agile Manifesto and the 12 Agile Principles presented in the subsequent section.

3.2.2 Twelve agile principles

The Agile Manifesto's authors also agreed on 12 Agile principles. You can use these principles to make sure your approach is true to agile values [2, 3]:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.
4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity, the art of maximizing the amount of work not done, is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, and then tunes and adjusts its behavior accordingly.

3.2.3 Comparison with traditional methodologies

Agile methodologies differ significantly from traditional models in how they handle project flow, customer involvement, and risk management. Table 3.1 provides a side-by-side comparison to highlight these differences.

While Waterfall, V-Model, and Spiral Model offer structure for projects with clearly defined processes and outputs, agile techniques are better suited to dynamic projects with requirements that may change.

3.2.4 Benefits of agile in software development

Agile development offers several advantages that make it highly effective in today's fast-paced and ever-changing software industry. These benefits contribute to better product quality, faster delivery, and increased customer satisfaction. The following are key benefits that can be identified:

- **Faster development cycles:** Agile methodologies focus on iterative development, allow-

Table 3.1: Comparison of software development methodologies.

Feature	**Agile**	**Waterfall**	**V-Model**	**Spiral Model**
Approach	Iterative & Incremental	Sequential & Linear	Verification & Validation	Iterative with Risk Analysis
Flexibility	High, adapts to changes	Low, rigid planning	Moderate, structured phases	High, adapts based on risks
Delivery	Continuous small releases	Single final release	Deliverables at each phase	Multiple incremental releases
Customer Involvement	High, frequent feedback	Low, feedback at the end	Low, involved in early stages	High, feedback in each iteration
Risk Management	Early detection of risks	Risks identified late	Moderate, addressed in testing	Strong, assessed at each phase
Documentation	Minimal, just enough	Extensive upfront planning	High, required at each phase	High, essential for risk tracking

ing teams to build and refine software in short sprints. This leads to faster time-to-market and the ability to release functional updates frequently.

- **Greater flexibility:** Agile's adaptability allows teams to accommodate changing project requirements even in later stages of development. This flexibility is crucial in dynamic industries where customer needs and market trends evolve rapidly.
- **Improved collaboration:** Agile encourages close collaboration between developers, stakeholders, and customers through regular meetings, such as daily stand-ups and sprint reviews. This transparency ensures alignment with business goals and enhances teamwork.
- **Higher customer satisfaction:** Continuous feedback loops enable teams to refine and improve the product based on real user needs. By involving customers throughout the process, Agile ensures that the final product aligns with expectations and provides value.
- **Risk reduction:** Frequent testing, incremental updates, and continuous integration help identify and resolve issues early. This proactive approach reduces the likelihood of major failures and ensures better-quality software.

3.3 Scrum framework

3.3.1 What is Scrum?

Scrum is a framework based on empiricism, meaning people who employ the scrum framework gain knowledge from real-life experience and make decisions based on that experience. It's a way of organizing your work. Scrum enhances collaboration, transparency, and adaptability while ensuring continuous improvement [4].

If you need to get something done, scrum provides a structure for increased efficiency and more effective results.

3.3.2 Scrum Roles

Scrum defines three essential roles that must be performed by the specific personnel engaged in project development: the product owner, the scrum master, and the development team.

Each of these roles is described below [4, 5].

Product owner

This person is responsible for laying out the work that needs to be done and prioritizing that work. They are knowledgeable in the project expectations and act as a guide to the team carrying out the project.

Product owners remain involved with the project throughout its entirety. This is as opposed to a boss who is only involved at the beginning, during initial planning. The Product Owner ensures that the project is adjusted and reprioritized as needed, based on feedback.

With all those duties, the product owner does not direct or control the development team's activities that is handled by the Scrum Master.

Scrum Master

The scrum master has two major responsibilities:

1. They protect the development team and make sure that everyone can focus on their work with minimal distractions. This includes isolating and handling impediments that come up.

2. They protect the scrum process itself and ensure that it is correctly applied. Therefore, the scrum master must be knowledgeable in agile and scrum. An element of this is acting as a trainer and coach for all other team members.

Development team

These are the individuals that write the code, The development team includes architects, testers, developers, and designers. The team acts together to figure out how to achieve their goals. The specific features they work on are based on the priority laid out by the product owner.

3.3.3 Scrum artifacts

Scrum's artifacts represent work or value. They are designed to maximize transparency of key information. Thus, everyone inspecting them has the same basis for adaptation. Each

artifact contains a commitment to ensure. It provides information that enhances transparency and focus against which progress can be measured [5]:

- For the *Product Backlog* it is the product goal.
- For the *Sprint Backlog* it is the sprint goal.
- For the *Increment* it is the definition of done.

These commitments exist to reinforce empiricism and the Scrum values for the Scrum team and their stakeholders.

Product backlog

The product backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the scrum team. Product backlog items that can be done by the scrum team within one sprint are deemed ready for selection in a sprint planning event. They usually acquire this degree of transparency after refining activities. Product backlog refinement is the act of breaking down and further defining product backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order, and size. Attributes often vary with the domain of work.

The developers who will be doing the work are responsible for the sizing. The product owner may influence the developers by helping them understand and select trade-offs.

Sprint Backlog

Sprint backlog is composed of the sprint goal (why), the set of product backlog items selected for the sprint (what), as well as an actionable plan for delivering the increment (how).

The sprint backlog is a plan by and for the developers. It is a highly visible, real-time picture of the work that the developers plan to accomplish during the sprint in order to achieve the sprint goal. Consequently, the sprint backlog is updated throughout the sprint as more is learned. It should have enough detail that they can inspect their progress in the daily scrum.

Increment

An increment is a concrete stepping stone toward the product goal. Each increment is additive to all prior increments and thoroughly verified, ensuring that all increments work together. In order to provide value, the increment must be usable.

Multiple increments may be created within a sprint. The sum of the increments is presented at the sprint review thus supporting empiricism. However, an increment may be delivered to stakeholders prior to the end of the sprint. The sprint review should never be considered a gate to releasing value. Work cannot be considered part of an increment unless it meets the definition of Done.

3.3.4 Scrum Events

In addition to these backlogs, scrum defines a series of events that are essential to the development process, namely [4, 5]:

1. **Sprint planning:**
    - Held at the start of each sprint to define objectives and select backlog items.
    - The team collaborates to estimate effort and break down tasks.
2. **Daily Stand-ups (Daily Scrum):**
    - Short, time-boxed meetings (usually 15 minutes) where team members share updates.
    - Focuses on progress, obstacles, and next steps.
3. **Sprint review:**
    - Conducted at the end of each sprint to showcase completed work.
    - Stakeholders provide feedback and discuss potential changes.
4. **Sprint retrospective**
    - A meeting for the scrum team to reflect on the sprint process.

- Identifies improvements for future sprints.

3.4 Modeling of use cases and interaction scenarios

3.4.1 Definition of a use case diagram

A use case diagram is a visual representation of a system's functionality from the user's perspective. It helps to identify the interactions between users (actors) and system functions (use cases). These diagrams are widely used in software engineering and system analysis to model system behavior [6].

3.4.2 Key components of a use case diagram

In what follows, the main components used to create a use case diagram are presented, namely [1, 6]:

- **Actors:**
  - Represent users or other systems that interact with the system.
  - Can be primary (initiates interaction) or secondary (supports interaction).
  - *Example:* A "Customer" using an "Online Shopping System".
- **Use cases:**
  - Represent actions or services the system provides.
  - Depicted as ovals inside the system boundary.
  - *Example:* "Place Order," "Make Payment."
- **System Boundary:**
  - Defines the scope of the system.
  - Represented as a rectangle enclosing the use cases.
- **Relationships:**

- **Association:** Links actors to use cases (solid line).
- **Include (`«include»`):** Represents mandatory functionality that is always executed within another use case.
- **Extend (`«extend»`):** Represents optional behavior that extends a base use case under specific conditions.
- **Generalization:** Represents inheritance among actors or use cases.

3.4.3 Benefits of use case diagrams

The use case diagrams offer numerous advantages, including [6]:

- Helps in requirement gathering and defining system scope.
- Provides clear communication between stakeholders.
- Supports identification of system functionalities and user interactions.
- Acts as a foundation for system design and documentation.

3.4.4 Presentation of our platform's use case diagrams

This section outlines the various use case diagrams relevant to the primary actors involved in our project:

- The stranded motorist (figure 3.1).
- The autonomous repairman (figure 3.2).
- The workshop administrator (figure 3.3).
- The workshop repairman (figure 3.4).
- The platform administrator (figure 3.5).

3.4.4.1 Use case diagram for the stranded motorist

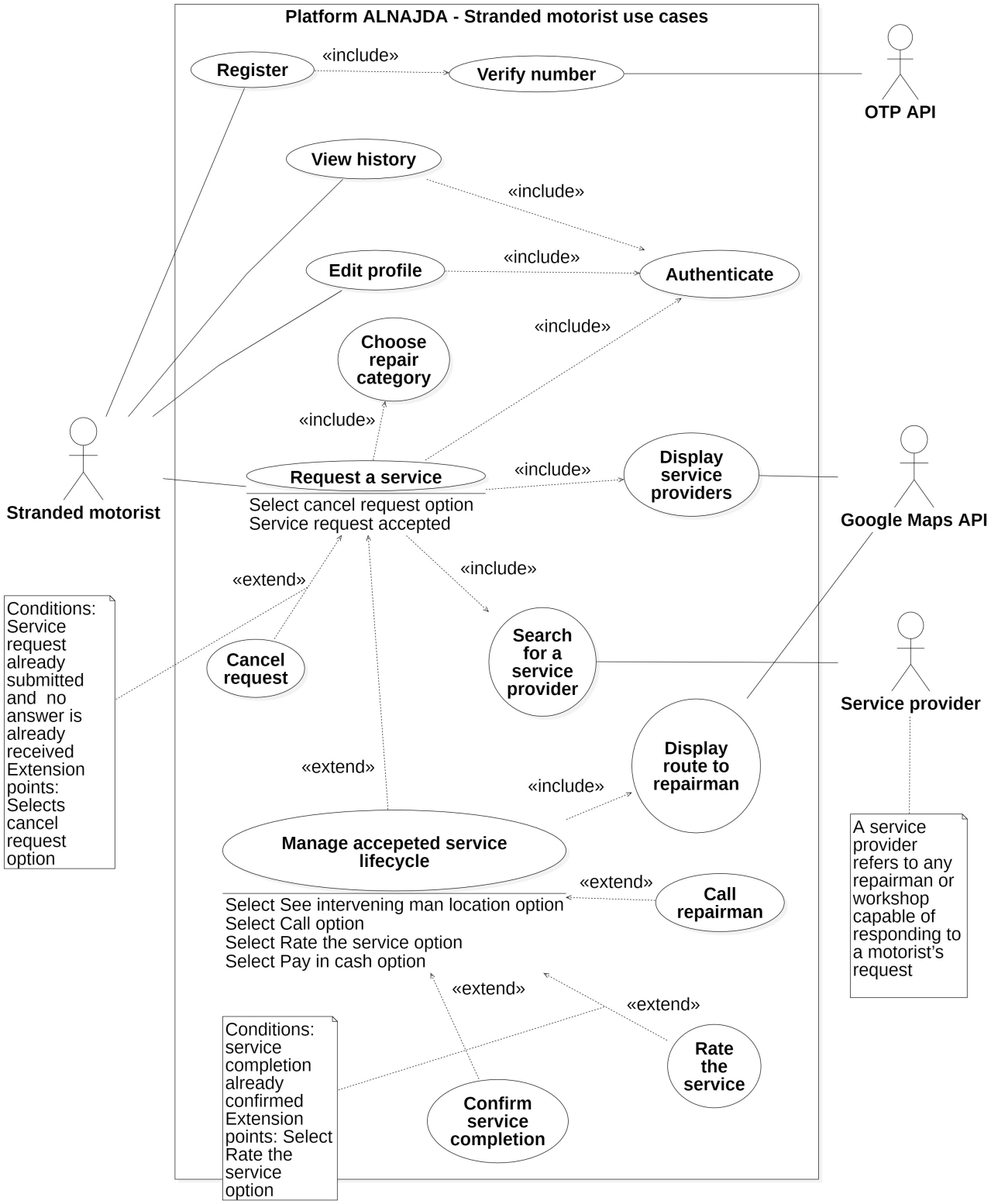

Figure 3.1: Our use case diagram for the stranded motorist.

3.4.4.2 Use case diagram for the autonomous repairman

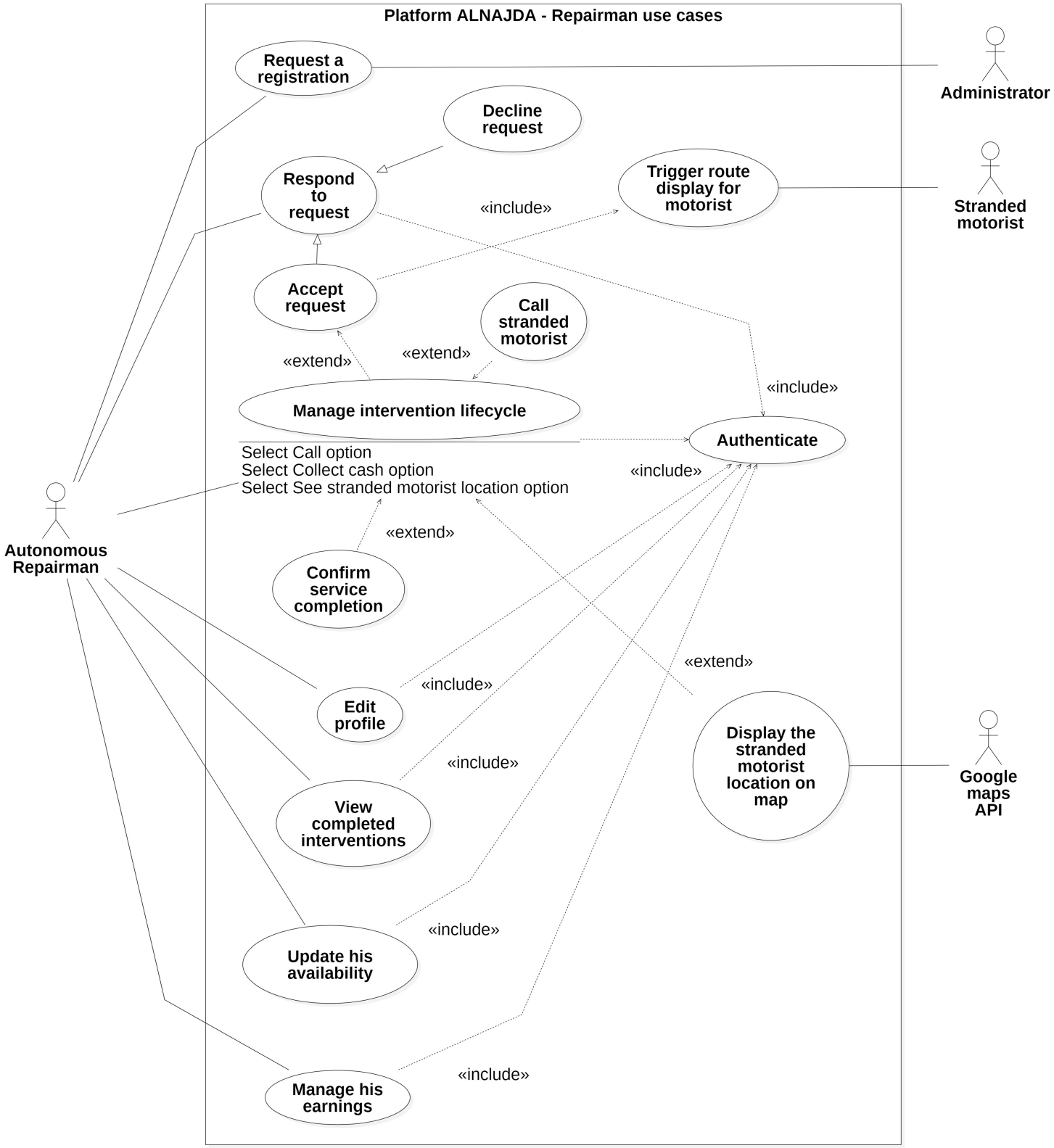

Figure 3.2: Our use case diagram for the autonomous repairman.

3.4.4.3 Use case diagram for the workshop administrator

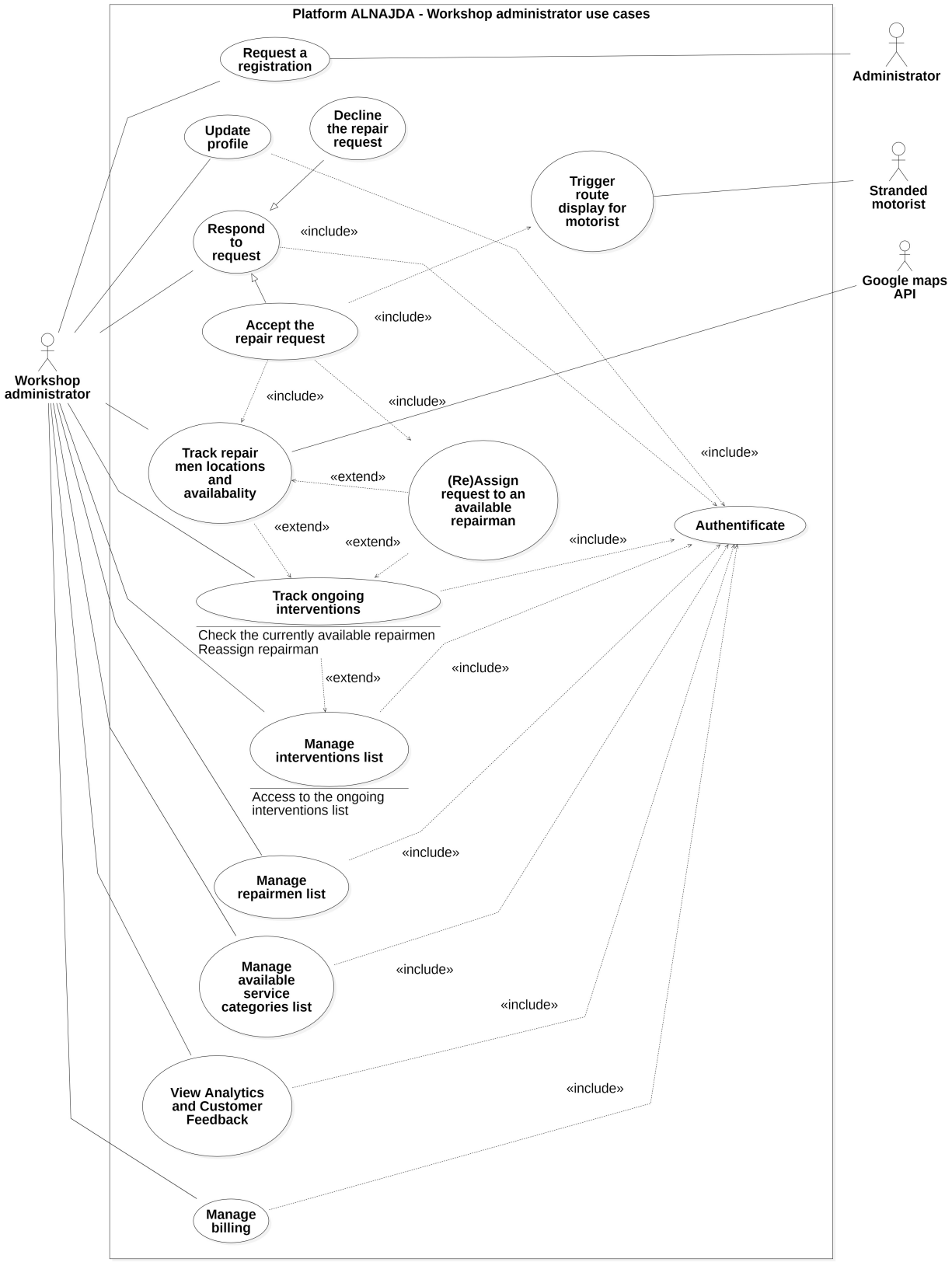

Figure 3.3: Our use case diagram for the workshop administrator.

3.4.4.4 Use case diagram for the workshop repairman

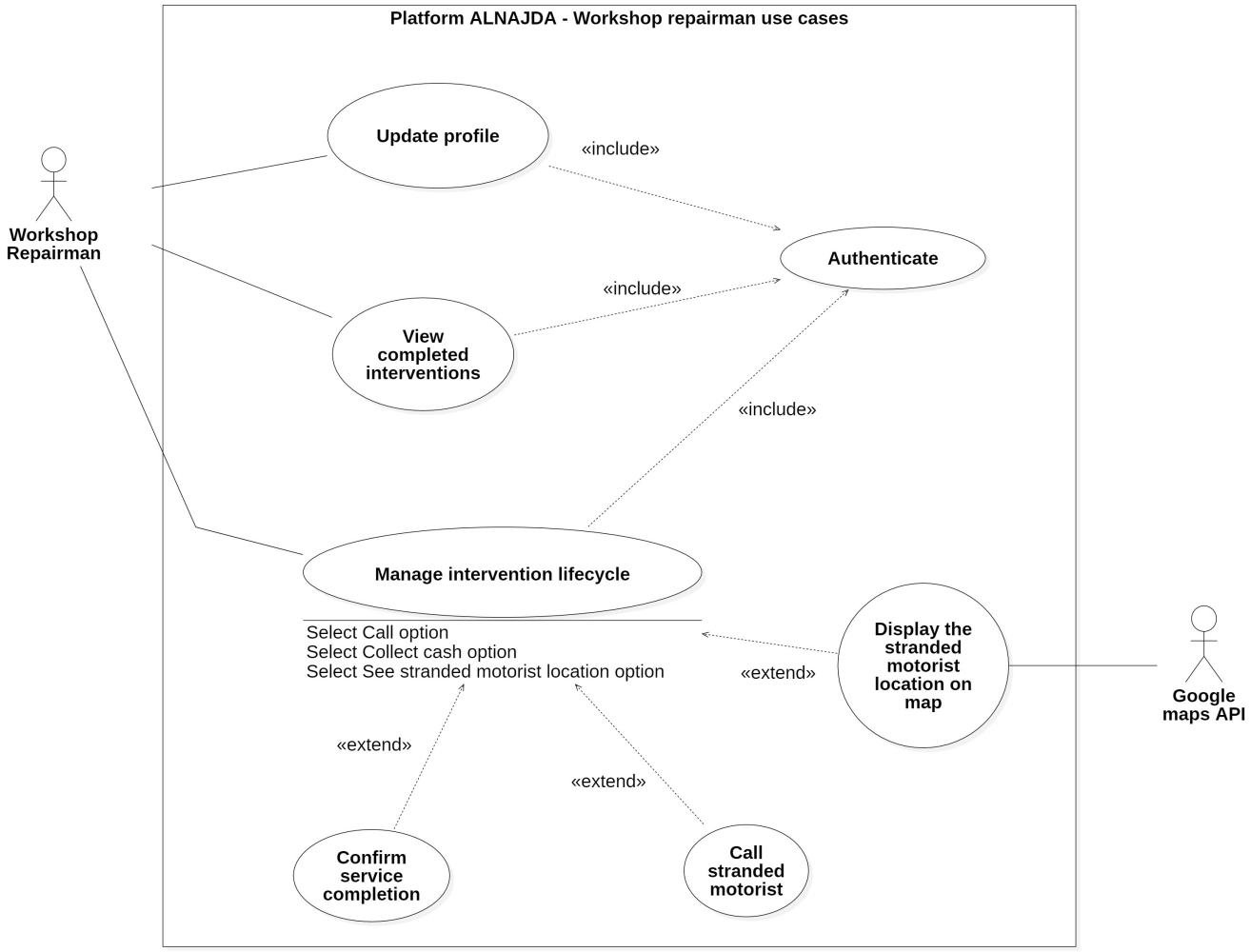

Figure 3.4: Our use case diagram for the workshop repairman.

3.4.4.5 Use case diagram for the plate-forme administrator

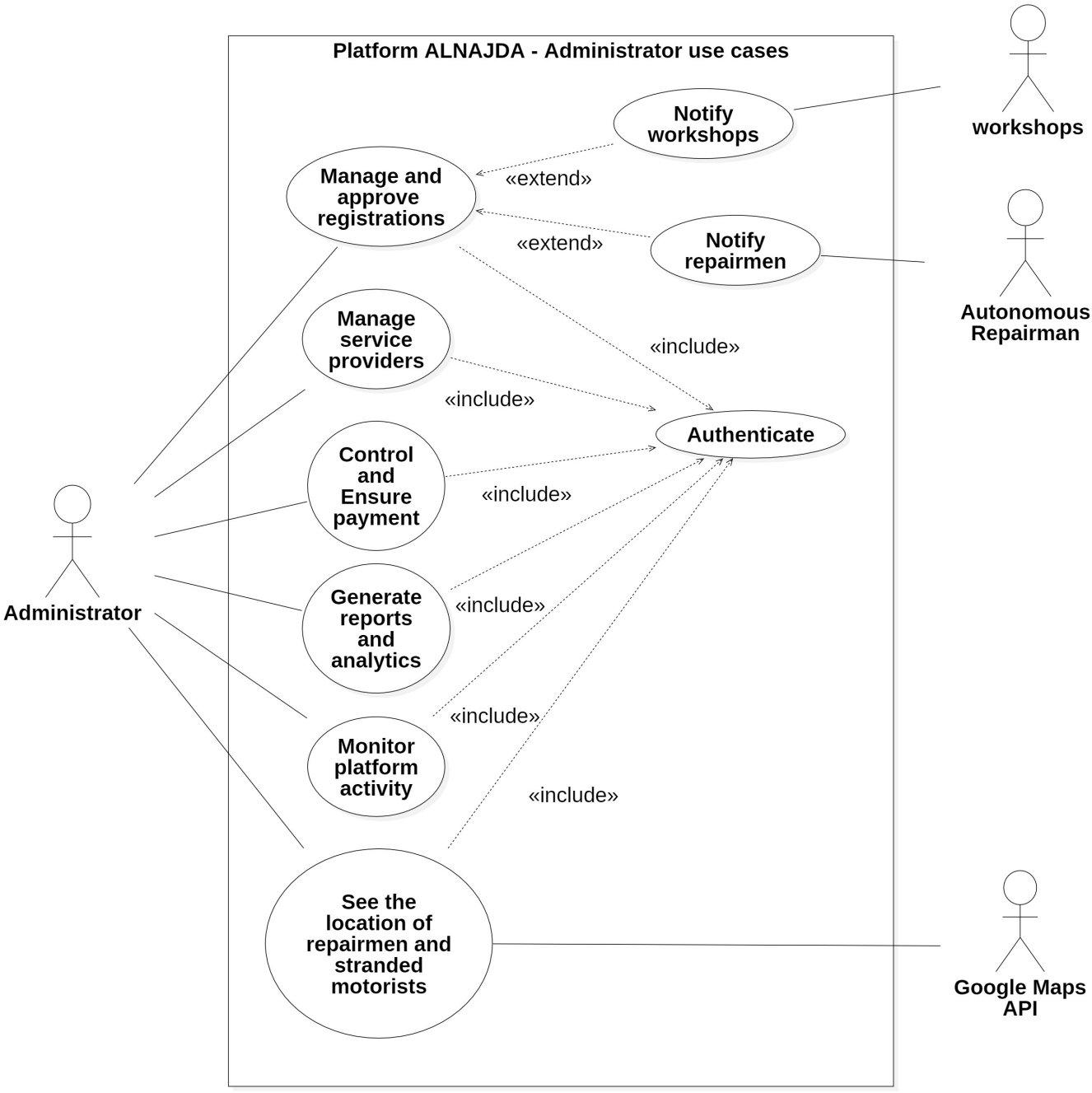

Figure 3.5: Our use case diagram for the plate-forme administrator.

3.4.5 Sequence diagrams for system-level interactions

This section presents several sequence diagrams illustrating system-level interactions for selected use cases. These diagrams, commonly referred to as System Sequence Diagrams (SSDs), depict the interactions between the system and the involved actors to achieve the intended functionality of each use case.

In what follows, we present the SSD corresponding to the use case *Request a Service*, along with the SSDs of the related included or extended use cases. The goal is to illustrate the complete implemented process required to request a service by the stranded motorist, from the initiation of the operation to the confirmation of service completion. Specifically, we present the SSDs corresponding to the following use cases:

- Request a service (figure 3.6).
- Authenticate (figure 3.7).
- Search for a service provider (figure 3.8).
- Manage accepted service lifecycle (figure 3.9).

3.4.5.1 SSD for the use case: Request a service

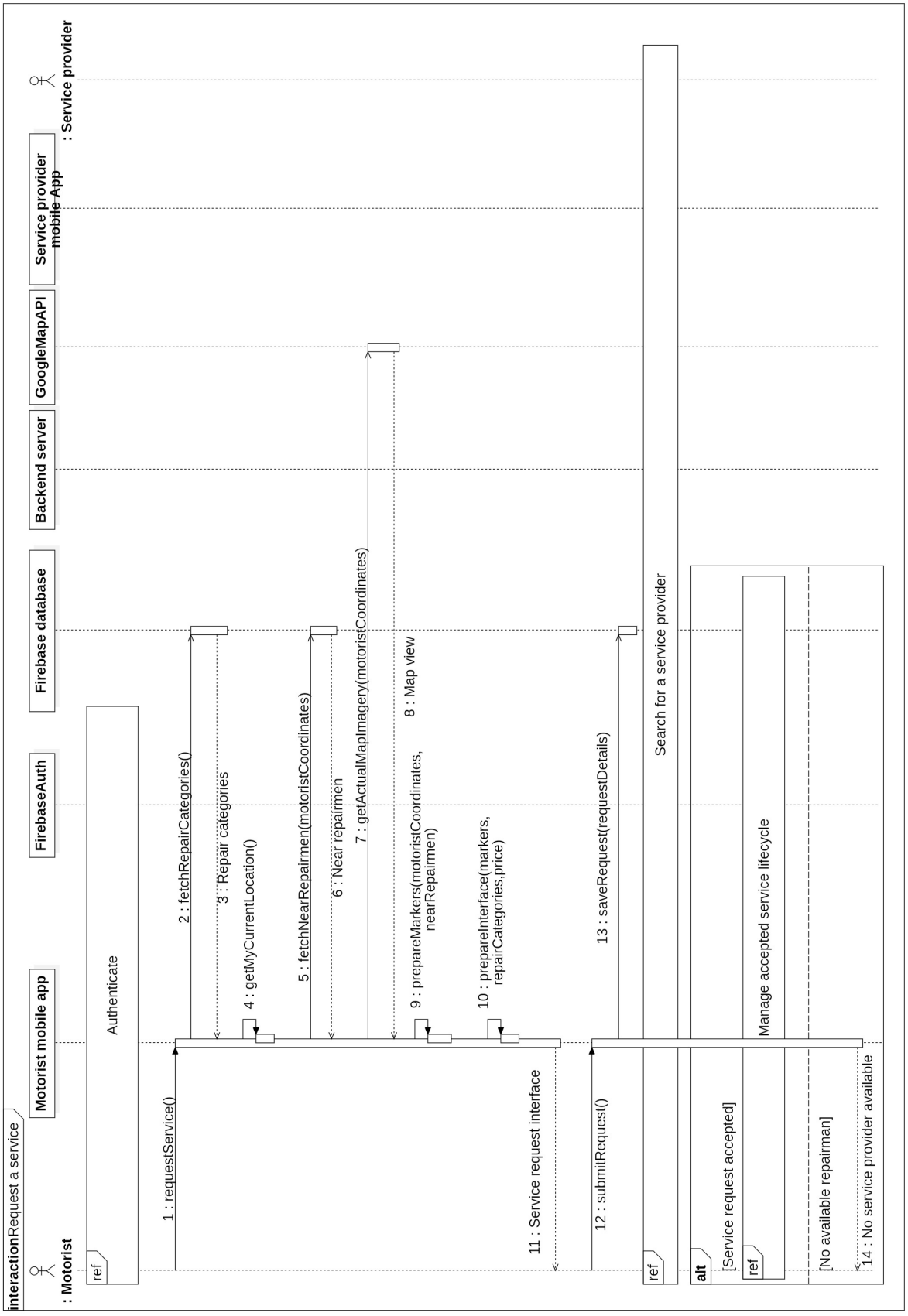

Figure 3.6: SSD for the use case: Request a service.

3.4.5.2 SSD for the use case: Authenticate (Stranded motorist)

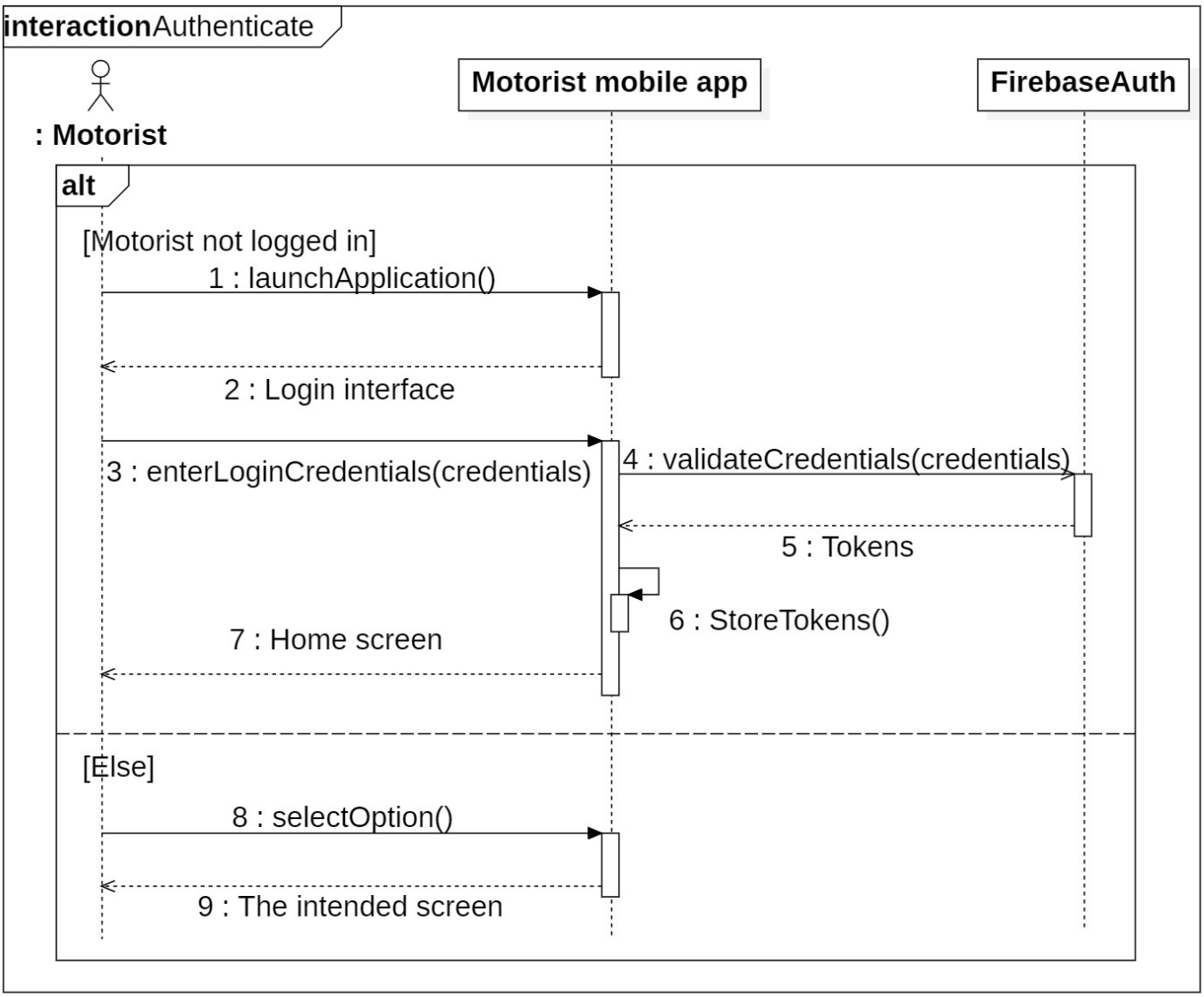

Figure 3.7: SSD for the use case: Authenticate.

3.4.5.3 SSD for the use case: Search for a service provider

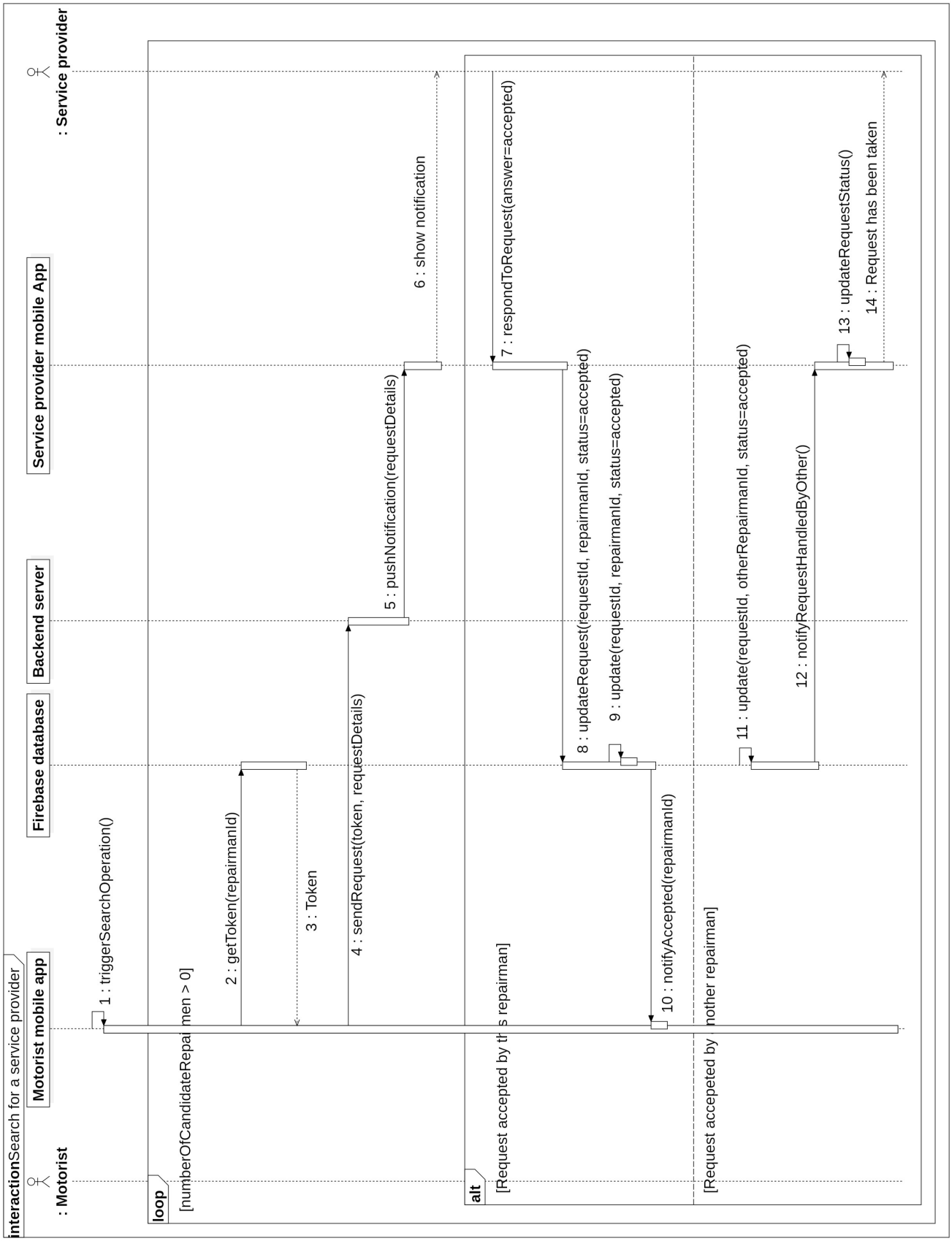

Figure 3.8: SSD for the use case: Search for a service provider.

3.4.5.4 SSD for the use case: Manage accepted service lifecycle

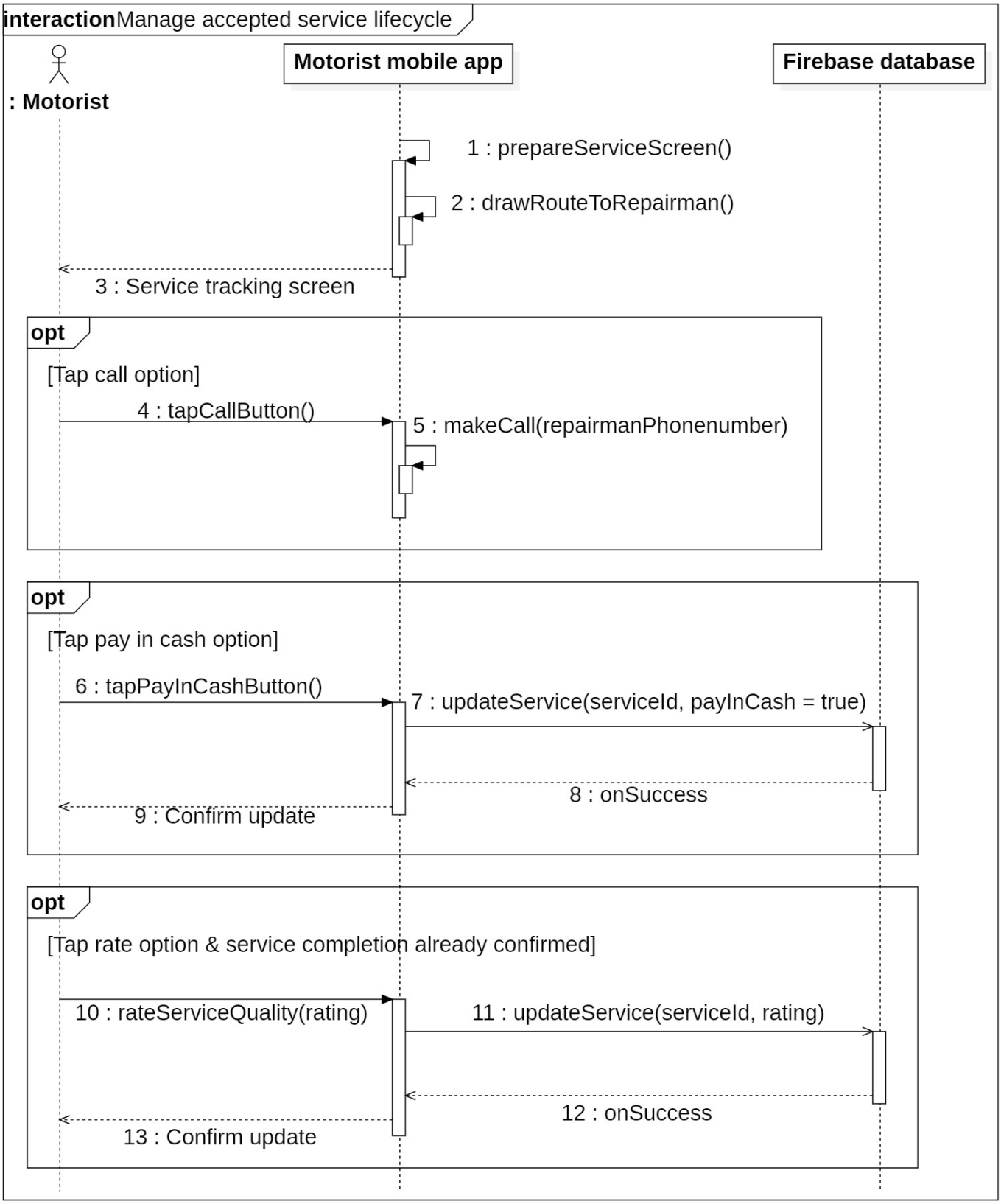

Figure 3.9: SSD for the use case: Manage accepted service lifecycle.

3.5 Scrum process for our development

This section outlines the application of the scrum process to realize our project objectives. It defines the assignment of scrum roles, specifies the established product backlog, identifies the defined sprints, and details the numerous conducted events.

3.5.1 Scrum roles identification

For our roadside assistance platform, we have assigned scrum roles to team members to ensure effective collaboration, transparency, and delivery of a user-centric solution. As previously mentioned, the scrum framework defines three core roles: *product owner*, *scrum master*, and *development team*. Below, we outline the team members, their assigned scrum roles, and their responsibilities within the context of the project. Stakeholders, while not formal scrum roles, are also noted for their critical input.

3.5.1.1 Product owner

- **Assigned to**: Rahim Oussama
- **Responsibilities**:
  - Represents stakeholders, including stranded motorists, autonomous repairmen, workshop administrators, workshop repairmen, and platform administrators.
  - Maintains and prioritizes the *product backlog*, ensuring user stories (e.g., US 1: Register with phone verification, US 4: Request service) align with the platform's goal of efficient roadside assistance.
  - Validates deliverables during *sprint reviews* to confirm features, such as service request notifications (US 6) and intervention lifecycle management (US 7, US 16, US 29), meet user needs.
  - Incorporates stakeholder feedback, such as motorist preferences for map-based provider displays (US 3), to refine requirements.
  - Acts as the primary liaison between the development team and stakeholders to ensure clear communication of business objectives.

3.5.1.2 Scrum master

- **Assigned to**: Our supervisor.
- **Responsibilities**:
  - Facilitates scrum events, including *sprint planning*, *daily stand-ups*, *sprint reviews*, and *sprint retrospectives*, ensuring adherence to scrum principles and time-boxing.
  - Removes impediments, such as technical blockers (e.g., API integration issues for US 3) or external distractions, enabling the development team to focus on delivering increments.
  - Coaches the team on agile and scrum practices, fostering a culture of continuous improvement through retrospectives and knowledge-sharing sessions.
  - Supports the product owner in backlog refinement, helping to break down complex user stories (e.g., US 7: Manage service lifecycle) into actionable tasks.

3.5.1.3 Development team

- **Assigned to**:
  - Rahim Oussama (Fullstack web developer)
  - Saidani Mohamed saleh Eddine (Mobile developer)
- **Responsibilities**:
  - Collaborates as a cross-functional unit to deliver potentially shippable increments, such as secure registration systems (US 1), map-based provider displays (US 3), and payment processing (US 34).
  - Estimates effort for user stories during *sprint planning*, using techniques like planning poker (e.g., 5 points for US 1, 10 points for US 7).
  - Designs, develops, tests, and deploys features, ensuring they meet the *definition of done* (e.g., code reviewed, unit tested, deployed to staging, documented).
  - Participates in *daily stand-ups* to report progress, discuss challenges (e.g., integrat-

ing geospatial APIs for US 32), and coordinate tasks.

- Contributes to backlog refinement, providing technical insights to split user stories (e.g., separating US 4's service request submission from destination selection for towing).

3.5.1.4 Stakeholders (Non-scrum roles)

While not part of the scrum team, the following stakeholders provide critical input and feedback:

- **Stranded motorist**: Represented by beta testers and user groups, they validate features like service requests (US 4), notifications (US 6), and service history (US 8) to ensure usability.
- **Autonomous repairman**: Consulted during *sprint reviews* to ensure intervention management (US 16) and availability updates (US 14) meet operational needs.
- **Workshop administrator**: Provides requirements for managing repairmen (US 24) and interventions (US 23), ensuring administrative efficiency.
- **Workshop repairman**: Tests intervention lifecycle features (US 29) to confirm practical applicability in field scenarios.
- **Administrator**: Offers feedback on platform oversight features, such as registration approvals (US 30) and analytics (US 35), to ensure robust governance.

By clearly defining these roles, we establish a collaborative framework where the scrum team works closely with stakeholders to deliver a high-quality roadside assistance platform that meets diverse user needs.

3.5.2 Product backlog with user stories

This section outlines the product backlog, detailing the items to be done to attain our primary objective. This description is articulated through user stories. A *user story* describes a software feature from an end-user perspective [6]. It helps the development team understand what the user needs and why. In its most common format, the sentence must contain three

elements: the who (role), the what (functionality), and the why (benefit). It takes this form:

As a <who>, I want to <what> so that I <why>

Table 3.2 presents our product backlog.

Table 3.2: Our product backlog.

ID	**User Story**	**Priority**	**Effort**
US 1	As a stranded motorist, I want to register with phone number verification so that I can securely access the platform.	H	5
US 2	As a stranded motorist, I want to update my profile so that my information remains accurate.	M	3
US 3	As a stranded motorist, I want to display the list of service providers on a map so that I can easily see nearby assistance.	M	5
US 4	As a stranded motorist, I want to request a service by choosing a repair category, and submitting a request so that I can get help.	H	8
US 5	As a stranded motorist, I want to cancel my service request if needed so that I have flexibility.	M	3
US 6	As a stranded motorist, I want to receive a notification when my request is accepted so that I am informed.	H	9
US 7	As a stranded motorist, I want to manage the accepted service lifecycle... so that I can track and evaluate the assistance.	H	10
US 8	As a stranded motorist, I want to view my service history so that I can review past assistance.	M	5

US 9	As an autonomous repairman, I want to request registration so that I can join the platform.	H	5
US 10	As an autonomous repairman, I want to authenticate so that I can access to platform.	H	5
US 11	As an autonomous repairman, I want to respond to requests... so that I can manage my workload.	H	9
US 12	As an autonomous repairman, I want the stranded motorist to receive a notification... so that they are informed.	H	9
US 13	As an autonomous repairman, I want to update my profile so that my information stays current.	M	3
US 14	As an autonomous repairman, I want to update my availability... so that clients can know when I am available.	M	4
US 15	As an autonomous repairman, I want to view the list of completed interventions... so that I can track my work history.	M	5
US 16	As an autonomous repairman, I want to manage the intervention lifecycle... so that I can deliver effective assistance.	H	10
US 17	As a workshop administrator, I want to request registration so that I can join the platform.	H	5
US 18	As a workshop administrator, I want to authenticate so that I can access to platform.	H	5

US 19	As a workshop administrator, I want to update my profile so that my information remains accurate.	M	3
US 20	As a workshop administrator, I want to respond to service requests... so that I can manage client requests.	H	5
US 21	As a workshop administrator, I want the stranded motorist to receive a notification and assign a workshop repairman...	H	8
US 22	As a workshop administrator, I want to see repairmen locations and availability...	M	6
US 23	As a workshop administrator, I want to manage the list of interventions...	H	8
US 24	As a workshop administrator, I want to manage the list of repairmen...	H	8
US 25	As a workshop administrator, I want to manage the available service categories...	M	5
US 26	As a workshop administrator, I want to view analytics and customer feedback...	M	4
US 27	As a workshop repairman, I want to authenticate so that I can access to platform.	H	5
US 28	As a workshop repairman, I want to view my completed intervention list...	M	3
US 29	As a workshop repairman, I want to manage the intervention lifecycle... so that I can provide effective assistance.	H	10

US 30	As an administrator, I want to approve or decline registration requests...	H	7
US 31	As an administrator, I want to manage the list of service providers...	H	8
US 32	As an administrator, I want to see the locations of repairmen and stranded motorists...	H	8
US 33	As an administrator, I want to manage the list of service categories...	M	4
US 34	As an administrator, I want to control payment processes...	H	9
US 35	As an administrator, I want to generate reports and analytics...	L	3

3.5.3 Sprints identification

The following sprint plan organizes development tasks into iterations based on goals and effort estimates. Each sprint focuses on specific features and functionality to ensure incremental delivery and value.

Sprint 1: Core registration and profile management

Goal: Build the foundational features for user registration, profile management, and access control.

Related items:

- US 1 – Register with phone verification (Stranded Motorist)
- US 9 – Request registration (Autonomous Repairman)
- US 10 – Authenticate (Autonomous Repairman)
- US 17 – Request registration (Workshop Administrator)

- US 18 – Authenticate (Workshop Administrator)
- US 27 – Authenticate (Workshop Repairman)
- US 30 – Approve/decline registration (Administrator)
- US 2 – Update profile (Stranded Motorist)
- US 13 – Update profile (Autonomous Repairman)
- US 19 – Update profile (Workshop Administrator)

Total effort: 47 points

Sprint 2: Service provider and service category management

Goal: Enable administrators to manage service providers and service categories.

Related items:

- US 31 – Manage service providers (Administrator)
- US 33 – Manage service categories (Administrator)
- US 25 – Manage service categories (Workshop Administrator)

Total effort: 17 points

Sprint 3: Service Request and Notification System

Goal: Enable stranded motorists to request services and notify service providers.

Related items:

- US 3 – Display providers on map (Stranded Motorist)
- US 4 – Request service (Stranded Motorist)
- US 5 – Cancel request (Stranded Motorist)
- US 6 – Receive notification (Stranded Motorist)
- US 11 – Respond to requests (Autonomous Repairman)

- US 12 – Notify motorist on acceptance (Autonomous Repairman)
- US 20 – Accept/decline requests (Workshop Admin)
- US 21 – Notify and assign repairman (Workshop Admin)

Total effort: 56 points

Sprint 4: Service lifecycle management

Goal: Enable users to manage service lifecycle (tracking, communication, billing, completion).

Related items:

- US 7 – Manage service lifecycle (Stranded Motorist)
- US 16 – Manage intervention lifecycle (Autonomous Repairman)
- US 29 – Manage intervention lifecycle (Workshop Repairman)
- US 22 – View repairmen locations/availability (Workshop Admin)
- US 32 – View all locations (Administrator)

Total effort: 44 points

Sprint 5: Service history and analytics

Goal: Enable users to view history and generate analytics.

Related items:

- US 8 – View service history (Stranded Motorist)
- US 15 – View completed interventions (Autonomous Repairman)
- US 23 – Manage intervention list (Workshop Admin)
- US 24 – Manage repairmen list (Workshop Admin)
- US 26 – View analytics/feedback (Workshop Admin)

- US 28 – View completed list (Workshop Repairman)
- US 35 – Generate analytics (Administrator)

Total effort: 36 points

Sprint 6: Payment processing and final testing

Goal: Implement payment processing and finalize testing phases.

Related items:

- US 34 – Control payment (Administrator)
- Conduct end-to-end and user acceptance testing
- Resolve bugs and issues
- Prepare platform for deployment

Total effort: 9 points

3.5.4 Conducted scrum events

During our development process, we implemented the following Scrum events to ensure smooth progress and continuous improvement:

Sprint planning

Conducted at the beginning of each sprint to define sprint objectives and select backlog items. The main performed tasks:

- The Product Owner presents high-priority user stories.
- The development team estimates the effort required for each story.
- Tasks are broken down and assigned to team members to ensure clarity and responsibility.

Daily stand-ups

Short meetings held every morning (15 minutes) to keep the team aligned.

- Each team member shares:
  - What they accomplished yesterday,
  - What they plan to do today,
  - Any blockers or issues they're facing.
- The scrum master addresses impediments and facilitates resolutions.

Sprint review

Held at the end of each sprint to demonstrate completed work.

- Team presents completed features to stakeholders.
- Stakeholders provide feedback and suggest improvements.
- New insights and requirements discovered during the sprint are discussed.

Sprint retrospective

A reflective session conducted after the Sprint Review.

- Team evaluates what went well, what could be improved, and what actions to take.
- Focus is on continuous improvement, collaboration, and team dynamics.

3.6 Database Design

3.6.1 Introduction to NoSQL

NoSQL databases represent a category of database management systems that diverge from traditional relational database models by offering flexible, scalable, and high-performance solutions for handling large volumes of unstructured or semi-structured data [7]. Unlike relational databases, which rely on structured tables with predefined schemas, NoSQL databases support various data models, including document-based, key-value, column-family, and graph databases. This flexibility makes NoSQL databases particularly suitable for modern applications, such as our vehicle breakdown assistance platform, which requires handling diverse data

types (e.g., user profiles, service requests, real-time location data, and intervention records) and scaling to accommodate a growing user base.

For our vehicle breakdown assistance platform, we utilize Firebase Realtime Database[1], a NoSQL database provided by Google, which organizes data in a hierarchical JSON tree structure. Firebase Realtime Database excels in delivering real-time synchronization, enabling instantaneous data updates across connected clients via WebSocket connections. This capability supports low-latency interactions, such as real-time updates for user actions or notifications, critical for dynamic applications [8]. Its schema-less design allows developers to store and modify data without predefined structures, facilitating rapid prototyping and iterative refinements that align seamlessly with Agile development principles.

3.6.2 Database design process

The database design process for our vehicle breakdown assistance platform followed a structured approach to ensure alignment with the platform's functional and non-functional requirements. The process consisted of the following steps:

- **Requirement Analysis:** The process begins with gathering and analyzing requirements from stakeholders, including stranded motorists, repairmen, workshop administrators, and platform administrators. Through Agile practices like user story mapping and sprint planning, we identified key data needs, such as real-time location data, user profiles, and transaction records. Firebase Realtime Database's schema-less JSON structure allows flexibility in capturing these diverse requirements without rigid schemas, enabling rapid adaptation to evolving user stories.

- **Conceptual Design:** In this phase, we developed a high-level model of the platform's data entities and their relationships, guided by use case diagrams (e.g., US 1, US 6). Unlike relational databases, Firebase Realtime Database uses a hierarchical JSON tree, which simplifies modeling complex relationships, such as those between users and service requests. This phase focused on defining core data entities and their interactions, ensuring support for real-time updates and geospatial queries critical for proximity-based matching.

[1] https://firebase.google.com/docs/database

- **Logical Design:** The logical design translates the conceptual model into a structure compatible with Firebase Realtime Database's NoSQL architecture. We organized data into a JSON tree, leveraging Firebase's event-driven architecture to support real-time synchronization for features like notifications and status updates. Security rules were defined to enforce access control, ensuring data integrity across user roles. The schema-less nature of Firebase allowed us to iterate on the logical design during sprints, accommodating changes without costly schema migrations.

- **Physical Design:** We configured Firebase to ensure efficient storage and retrieval, including setting up indexes for frequently queried fields (e.g., user location, service status) and sharding to support scalability.

- **Testing and Validation:** The database schema was tested during development sprints to ensure it supported all user stories (e.g., US 3: Display providers on a map, US 7: Manage service lifecycle). Performance tests validated the database's ability to handle concurrent requests and scale with increasing user activity.

This process ensured that the database design was robust, scalable, and aligned with the platform's operational needs.

3.6.3 Our conceptual data model

Figure 3.10 illustrates our class diagram, which represents the system's data requirements. This class diagram serves as a conceptual data model (CDM), illustrating the essential data needed to ensure the proper implementation of the features offered on our website.

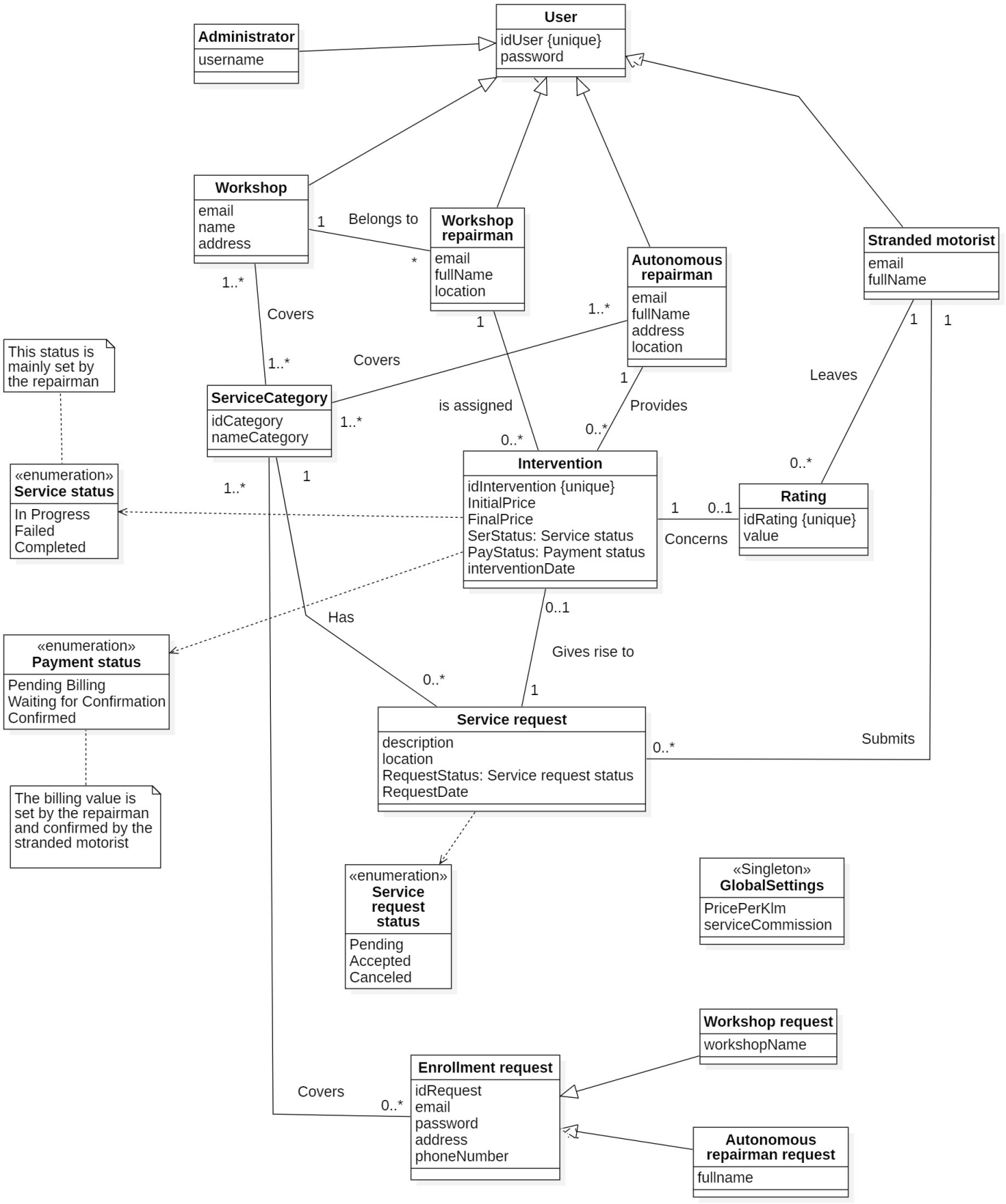

Figure 3.10: Conceptual data modeling using a class diagram.

3.6.4 Our logical data design

The database is structured into the following top-level nodes:

activeDrivers:

Stores real-time location data for active drivers, enabling geospatial queries for matching stranded motorists with nearby repairmen (US 3). Each entry is keyed by a driver's unique ID and includes fields like l (a GeoPoint array with latitude and longitude) and priority for sorting or ranking drivers based on availability or proximity.

Admin:

Contains credentials for platform administrators, including username and password, supporting secure access to administrative functions like registration approvals (US 30) and analytics (US 35).

repaimen:

Manages profiles for drivers (autonomous repairmen), with fields such as id (unique identifier), name, email, phone, address, car-details (subfields: vehicle-color, vehicle-model, vehicle-number, type), newRideStatus (e.g., idle, accepted), earnings, tripsHistory, token (for push notifications), and valide (validation status). This node supports repairmen management (US 14, US 16).

globalsettings:

Stores platform-wide configurations, such as price-per-km and service-commission, enabling dynamic pricing and commission calculations for payment processing (US 34).

registrationRequests:

Tracks workshop registration requests, with fields like createdAt, email, fullAddress, ownerFullname, phone, services (e.g., electrical, tire-service), status (e.g., approved), and identificationDocuments (subfields: fileName, fileType, url). This node supports registration approval workflows (US 30).

workshopRepairmen:

Stores data for repairmen associated with workshops, including uid (unique identifier), username, email, category (e.g., electrical, tire-service), status (e.g., active, blocked), createdAt, and workshopId. This node facilitates workshop repairmen management (US 24, US 29).

motorist:

Contains profiles for stranded motorists, with fields like id, name, email, phone, and address. This node supports user registration and authentication (US 1) and service request submission (US 4).

workshops:

Manages workshop data, with fields such as uid, workshopName, email, fullAddress, ownerFullname, phone, services, status, createdAt, and identificationDocuments. This node supports workshop management and service assignments (US 23, US 24).

The use of geospatial indexes on the location fields in the Users, ServiceRequests, and Workshops collections enables efficient querying for nearby technicians and realtime tracking. Embedded documents reduce the need for complex joins, improving performance for read-heavy operations, such as displaying service providers on a map . The schema is designed to be flexible, allowing for future additions, such as new service categories or user roles, without requiring significant restructuring .

3.7 Conclusion

This chapter presented the methodology and design approach for our vehicle breakdown assistance platform, emphasizing the use of Agile and Scrum to ensure iterative development and stakeholder collaboration. The adoption of Scrum facilitated structured sprints, clear role assignments (e.g., Rahim Oussama as Product Owner, Saidani Mohamed Saleh Eddine as a mobile developer), and continuous feedback through events like sprint reviews and retrospectives. Use case diagrams provided a clear visualization of user interactions, guiding the development of features for stranded motorists, technicians, workshop administrators, and platform administrators. The NoSQL database design, leveraging Firebase Realtime Database's JSON-based

model, ensures flexibility, real-time synchronization, and scalability, particularly for geospatial queries critical to the platform's core features. Together, these elements form a robust foundation for implementing a user-centric, efficient, and reliable roadside assistance platform.

The next chapter will detail the implementation phase, discussing the technologies used and solutions applied to realize the designed system.

Chapter 4

Implementation

4.1 Introduction

This chapter describes our car breakdown support platform, therefore transforming the architecture and approach described in Chapter 3 into a working solution. Through mobile apps and web interfaces, the platform seeks to meet the demands of stranded drivers, autonomous repairmen, workshop repairmen, workshop managers, and plat-form administrators by offering effective, open, user-centric roadside help. The Agile and Scrum models guided the iterative implementation to provide constant input and alignment with user needs. Backend API, user flow diagrams, important application interfaces, system architecture, technologies and tools are covered in this chapter.

4.2 Architecture of Our System

The system architecture is a full-stack application designed to support real-time, scalable, and secure roadside assistance services. It comprises the following components, which interact seamlessly to deliver the platforms functionalities.

Web Client (Next.js)

- Built using Next.js for server-side rendering and client-side interactivity, with TypeScript for type safety, Tailwind CSS for responsive styling, and ShadenUI for reusable compo-

nents.

- Handles user interactions for workshop administrators and platform administrators, communicating with the backend via REST API requests in JSON format.

Mobile Application

- Developed for Android and iOS, targeting stranded motorists, autonomous repairmen, and workshop repairmen.
- Interacts with the backend via REST API, enabling service request creation, real-time tracking, and intervention management.
- Uses native APIs for GPS integration to support location-based services.

Backend Services

- Implemented as a REST API using Node.js and Express.js, exposing endpoints (GET, POST, PUT, DELETE) for user management, service requests, and payments.
- Integrates with FireBase for data storage and retrieval, using JSON for data exchange.
- Handles business logic, such as technician matching and notification dispatching.

Database (Firebase)

- A NoSQL database utilizing Firebase Realtime Database's hierarchical JSON structure to store data for users, drivers, repairmen, workshops, registration requests, platform settings, and real-time location tracking.
- Supports real-time synchronization for instantaneous updates and geospatial data handling for efficient location-based queries critical to the platform's functionality.

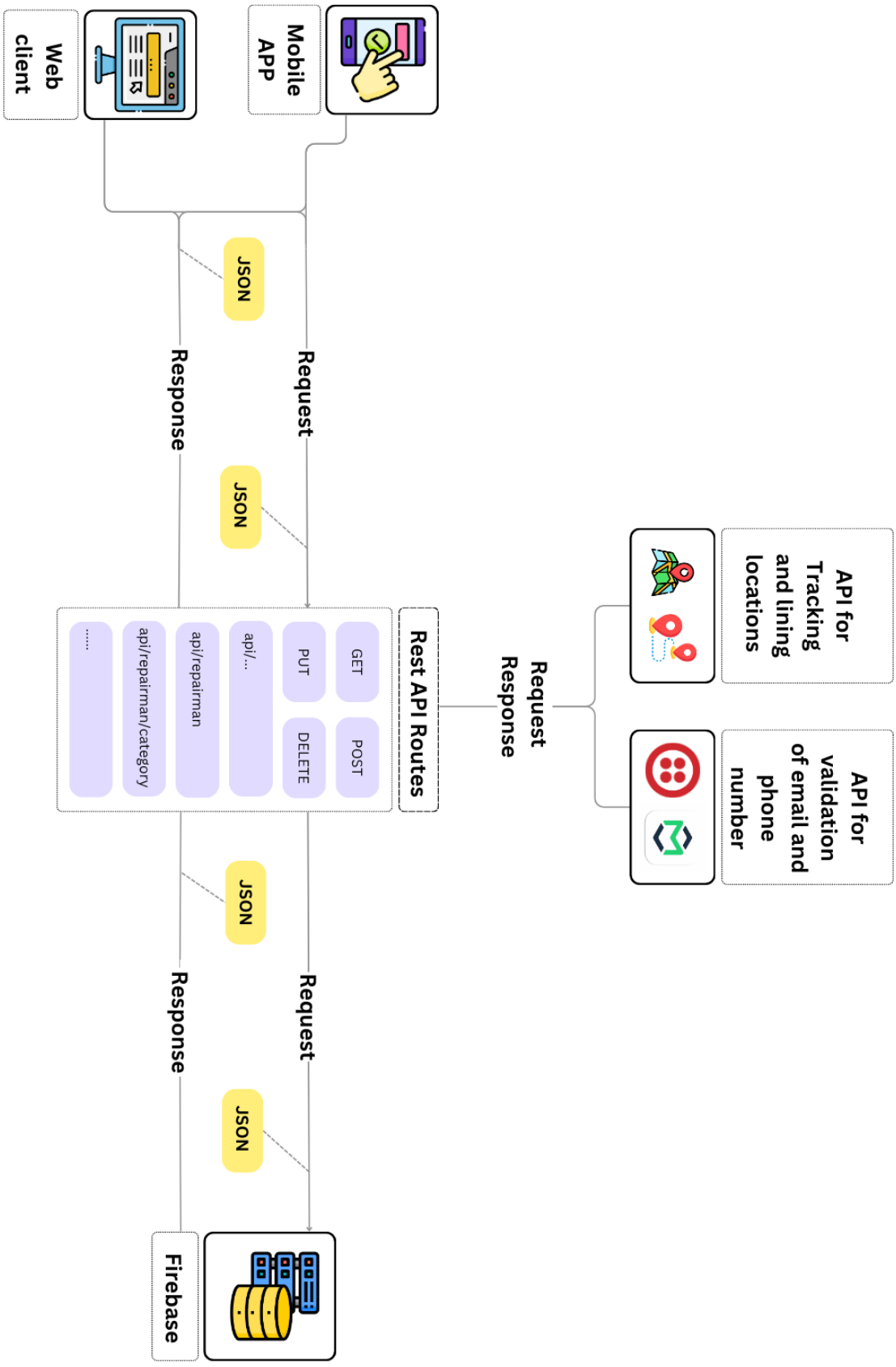

Figure 4.1: System architecture diagram.

Third-Party APIs

- **Tracking & Location API:** Supports real-time tracking of technicians and motorists.
- **Email & Phone Validation API:** Ensures secure user registration.

Figure 4.1 illustrates the system architecture, highlighting the interaction between the web client (Next.js), backend (REST API), Flutter database, mobile app, and third-party APIs for tracking and validation. The diagram shows REST API routes (e.g., GET, POST, PUT, DELETE) and data exchange in JSON format, reflecting the scalable and real-time nature of the implementation.

These components interact via REST API calls, with the backend orchestrating data flow between the web client, mobile app, database, and third-party services. The architecture ensures scalability, reliability, and performance, meeting the non-functional requirements.

4.3 Technologies, Tools, and Programming Languages

The implementation leveraged a modern technology stack to ensure scalability, maintain ability, and user-friendliness. The technologies and tools are grouped by their roles in the system:

- Technologies used at the front-End.
- Technologies used at the back-end.
- Development tools.
- Third-Party services.

In line with Agile principles and to fulfill the platform's functional objectives, the subsequent technological stack was chosen to provide quick iteration and scalability. Each aforementioned category is detailed in the subsequent sections.

4.3.1 Front-End

- **Next.js:** Next.js is a powerful, React-based, open-source framework developed by Vercel for building server-rendered and statically generated web applications. It simplifies the

development of modern web applications by offering features like server-side rendering (SSR), static site generation (SSG), and API routes, which enhance performance and SEO. Next.js supports file-based routing, automatic code splitting, and TypeScript out of the box, making it ideal for building scalable, production-ready applications [9].

- **TypeScript:** TypeScript is a statically typed superset of JavaScript, developed and maintained by Microsoft, designed to enhance JavaScript by adding type definitions. It improves code maintainability and catches errors during development, making it particularly useful for large-scale applications. TypeScript compiles to plain JavaScript, ensuring compatibility with any JavaScript environment [10].

- **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework that enables rapid UI development by providing a comprehensive set of pre-defined, low-level utility classes. Unlike traditional frameworks like Bootstrap, Tailwind allows developers to style elements directly in HTML or JSX without writing custom CSS, promoting flexibility and modularity. It's highly compatible with Next.js and supports features like Just-In-Time (JIT) compilation for optimized builds [11].

- **Shadcn UI:** Shadcn UI is an open-source UI library built on top of Tailwind CSS and Radix UI, designed for creating accessible, customizable, and minimal UI components. Unlike traditional component libraries, Shadcn UI provides copy-pasteable code that developers can fully own and modify, avoiding hidden abstractions. It's particularly popular for SaaS dashboards and Next.js applications due to its ease of use and TypeScript support [12].

- **FLutter:** Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Written in Dart, Flutter uses a widget-based architecture to create highly customizable, performant, and visually appealing interfaces [13].

4.3.2 Back-End

- **Node.js:** Node.js is an open-source, server-side runtime environment built on Chrome's V8 JavaScript engine, enabling developers to run JavaScript outside the browser. It is designed for building scalable, high-performance applications, particularly those requiring

asynchronous, non-blocking I/O operations. Node.js is widely used for backend development, APIs, and real-time applications [14].

- **Express.js:** Express.js is a minimal and flexible web application framework for Node.js, designed to simplify the creation of robust APIs and web servers. It provides a lightweight layer on top of Node.js, offering features like routing, middleware, and HTTP request handling, making it a go-to choice for backend development.

- **Firebase:** Firebase Realtime Database is a NoSQL, JSON-based database that stores data in a hierarchical tree structure, offering real-time synchronization and scalability for modern applications. Unlike relational databases, its schema-less design supports flexible handling of unstructured or semi-structured data, making it ideal for real-time features like location tracking and notifications.

4.3.3 Tools

- **Git:** Git is a distributed version control system designed for tracking changes in source code during software development. It enables multiple developers to collaborate on projects, manage code versions, and maintain a history of changes [15].

- **Docker:** Docker is an open-source platform for containerization, allowing developers to package applications and their dependencies into lightweight, portable containers. Containers ensure consistent environments across development, testing, and production.

- **Postman:** Postman is a popular API development and testing tool that simplifies designing, testing, and documenting APIs. It provides a user-friendly interface for sending HTTP requests, validating responses, and automating API workflows [16].

- **VS Code:** Visual Studio Code is a lightweight, open-source code editor developed by Microsoft, optimized for web and mobile development with support for JavaScript, TypeScript, Node.js, and more. It's highly customizable with extensions [17].

4.3.4 Third-Party Services

- **Google Maps API:** The Google Maps API is a set of APIs and services provided by Google that enables developers to integrate interactive maps and location-based features

into web and mobile applications. Primarily, the Maps JavaScript API is used for web applications, while other APIs like the Maps SDK for Android, iOS, and Flutter cater to mobile and cross-platform development. It allows developers to create dynamic maps, add markers, calculate routes, and leverage location-based services like Places, Geocoding, and Directions, enhancing user experiences with geospatial functionality [18].

- **NextAuth.js:** NextAuth.js (now transitioning to Auth.js) is an open-source authentication library designed for Next.js applications, simplifying the integration of secure authentication mechanisms like OAuth, email-based sign-in, and credentials-based login. It supports providers like Google, GitHub, and more, making it ideal for adding single sign-on (SSO) or custom authentication to web apps [19].

4.4 Presentation of Our API

This section describes the REST API that enables interaction between the mobile applications, web client, and Firebase database. The API, built with Node.js and Express.js, exposes endpoints to support key functionalities, such as user management, service requests, and payments. Each endpoint is designed to handle specific user stories (Section 3.5.2) and ensure secure, efficient data exchange in JSON format.

4.4.1 API Routes for Stranded Motorists (Mobile Application)

The API routes for stranded motorists are designed to support the full lifecycle of a road side assistance request, from authentication to payment and feedback. These endpoints are optimized for mobile usage, ensuring low latency and real-time updates through integration with the Google Maps API for location tracking. Figure 4.2 depicts our developed API that supports specific features for stranded motorists.

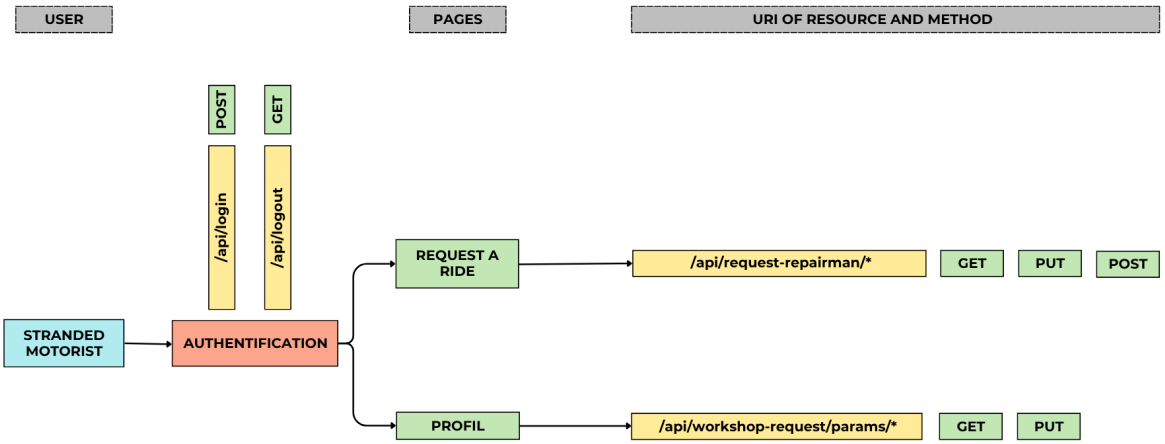

Figure 4.2: API Routes for Stranded Motorists.

4.4.2 API Routes for Autonomous Repairmen and Workshop Repairmen (Mobile Application)

The API routes for repairmen focus on managing availability, accepting requests, and updating intervention statuses. These endpoints are designed to support both autonomous repairmen (who seek their own jobs) and workshop repairmen (who receive assigned tasks). Real-time updates are achieved through WebSocket integration for notifications, and geospatial queries leverage Firebase Realtime Database's capabilities for efficient location-based matching, utilizing the hierarchical JSON structure to access real-time location data. Figure 4.3 emphasizes our developed API, which facilitates various functionalities for Autonomous Repairmen and Workshop Repairmen.

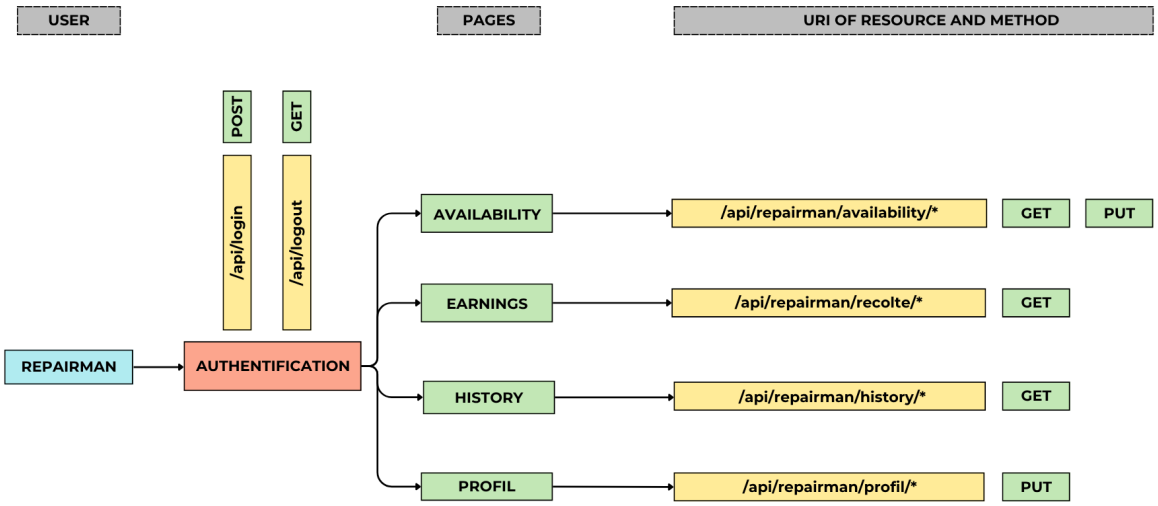

Figure 4.3: API Routes for Autonomous Repairmen and Workshop Repairmen.

4.4.3 API Routes for Workshop Administrators (Web Application)

Workshop administrators use API routes to manage repairmen, assign service requests, and monitor interventions. The details of the designed API are described in figure 4.4.

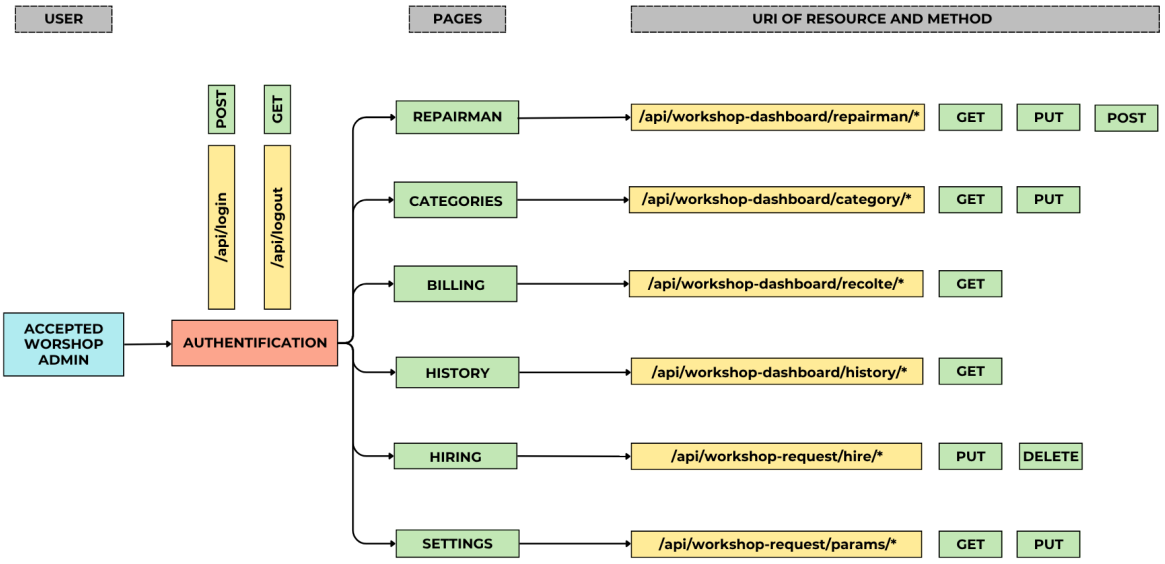

Figure 4.4: API Routes for Workshop Administrators.

4.4.4 API Routes for Platform Administrators (Web Application)

Platform administrators oversee the entire ecosystem, requiring API routes for user management, registration approvals, and system-wide analytics. These endpoints include advanced filtering and auditing capabilities, with logging implemented for compliance. Rate limiting is stricter to protect sensitive operations. The specifications of the implemented API are illustrated in Figure 4.5.

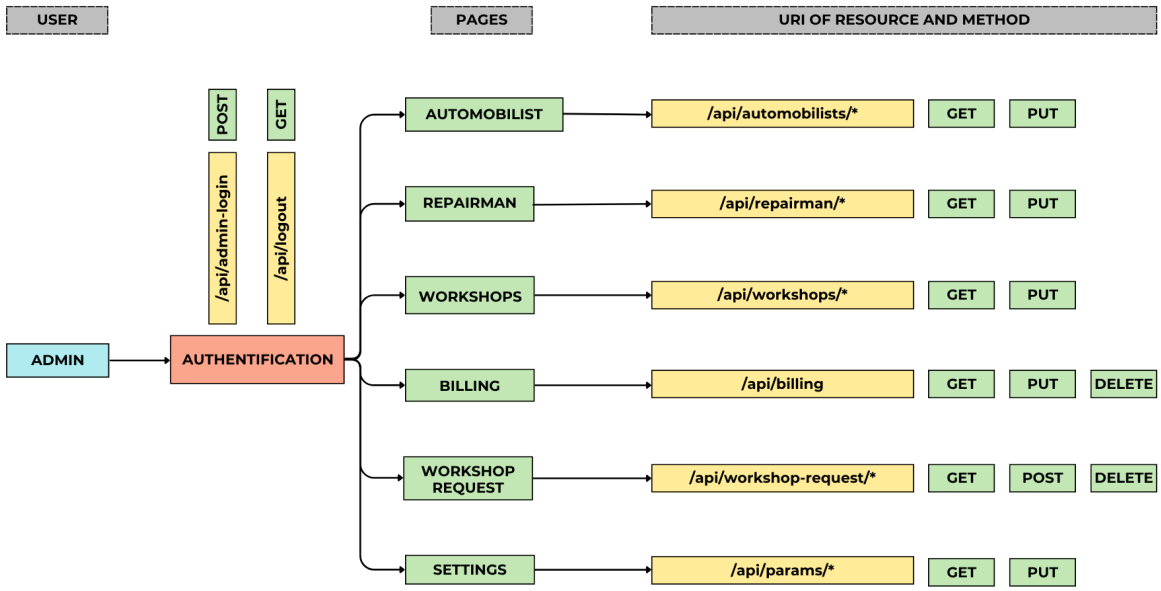

Figure 4.5: API Routes for Platform Administrators.

4.5 User Flow Diagrams

User flow diagrams illustrate the steps taken by each actor to achieve specific goals within the platform, depicting screen transitions and interactions. Below, we present flow diagrams for the primary actors, focusing on key functionalities.

4.5.1 Stranded Motorist Flow Diagram

The stranded motorist flow begins with account creation or login, followed by submitting a service request with a chosen breakdown category and location. The motorist can then view nearby providers, receive notifications, track the technician's progress, and complete payment

with feedback upon service completion.

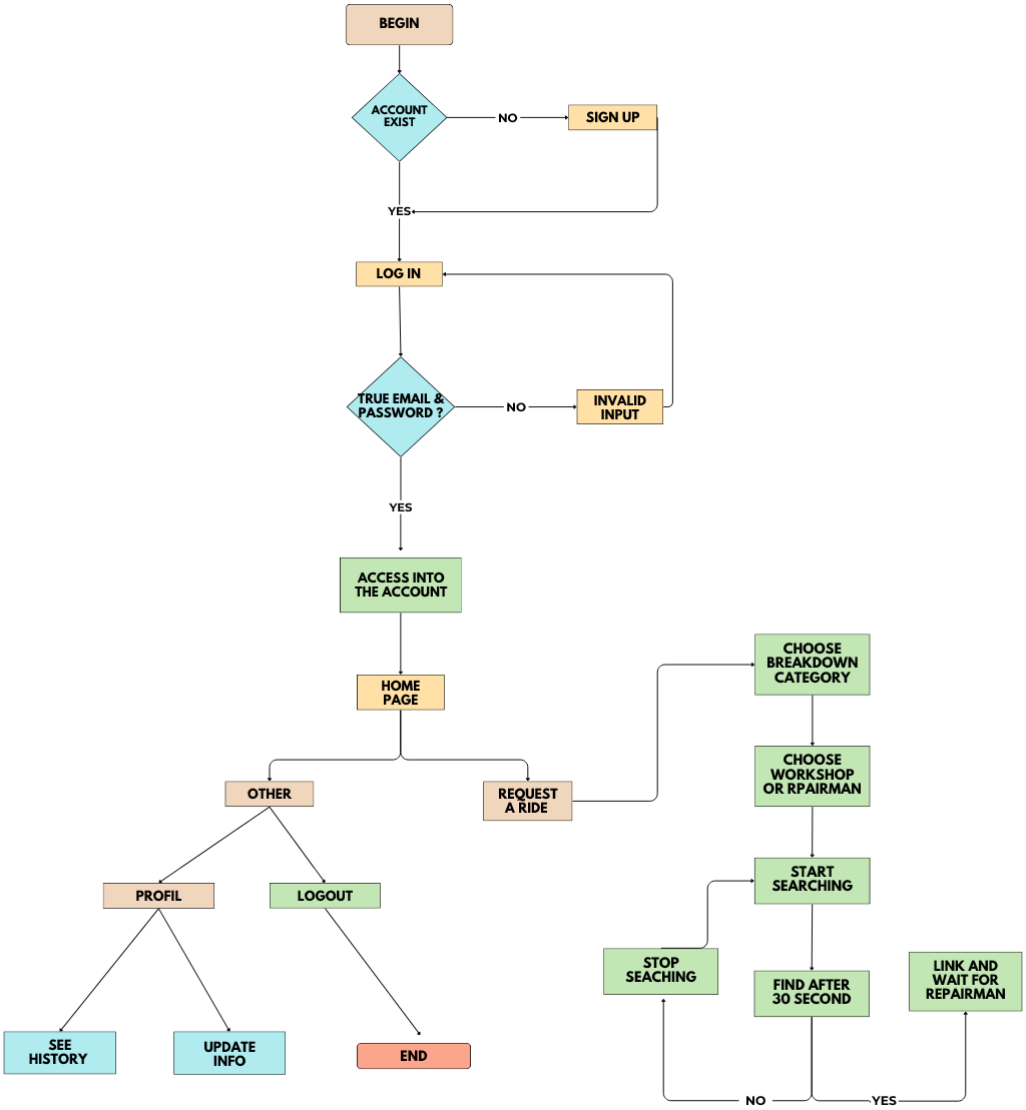

Figure 4.6: User flow diagram for stranded motorists.

4.5.2 Autonomous Repairman Flow Diagram

The autonomous repairman flow starts with login and account access, allowing them to update availability, view nearby service requests, accept or manage interventions, and review their earnings and intervention history for effective job management.

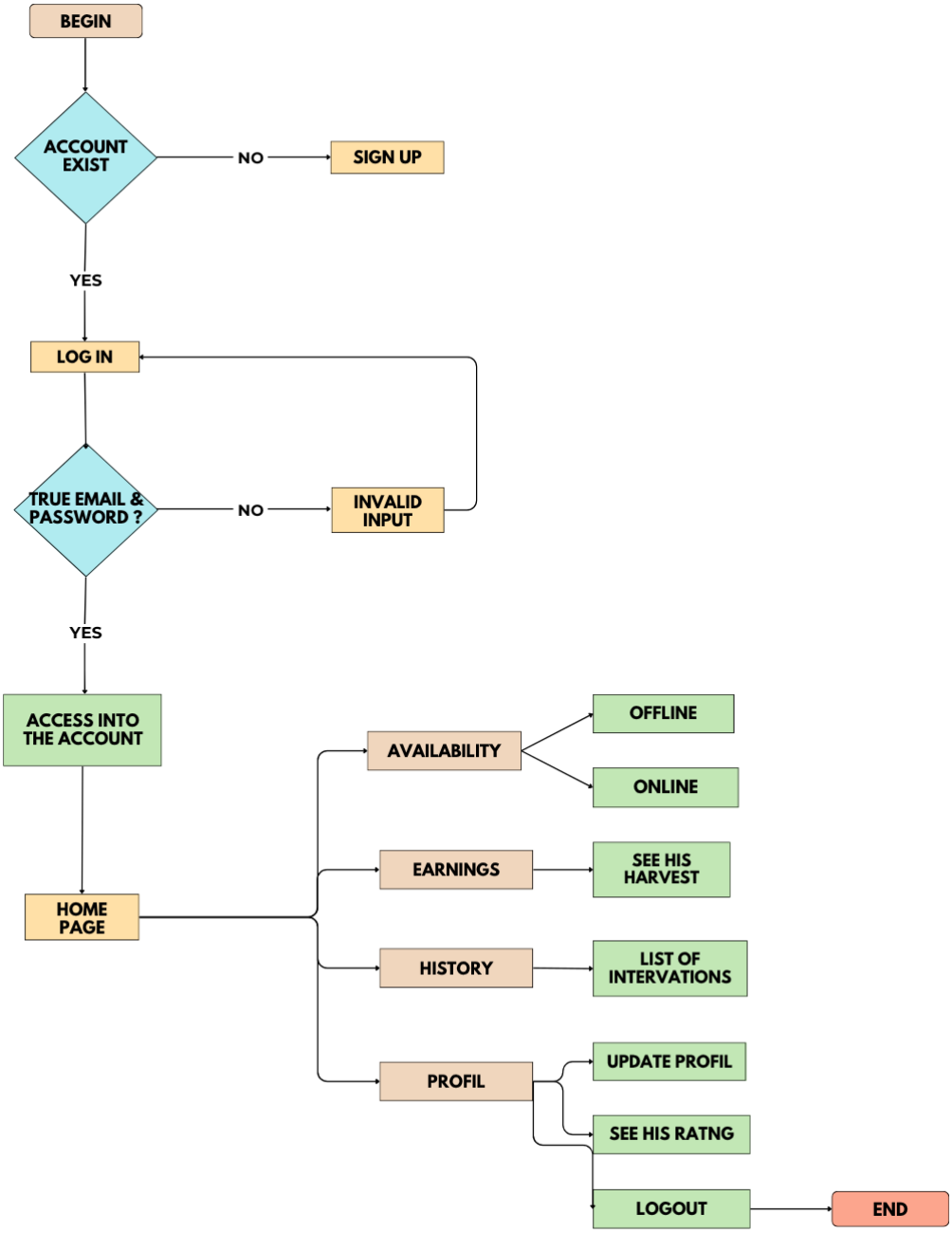

Figure 4.7: User flow diagram for autonomous repairmen.

4.5.3 Workshop Administrator Flow Diagram

The workshop administrator flow involves logging in to access a dashboard, where they can manage repairmen, assign service requests, track interventions, and analyze performance metrics to ensure efficient workshop operations.

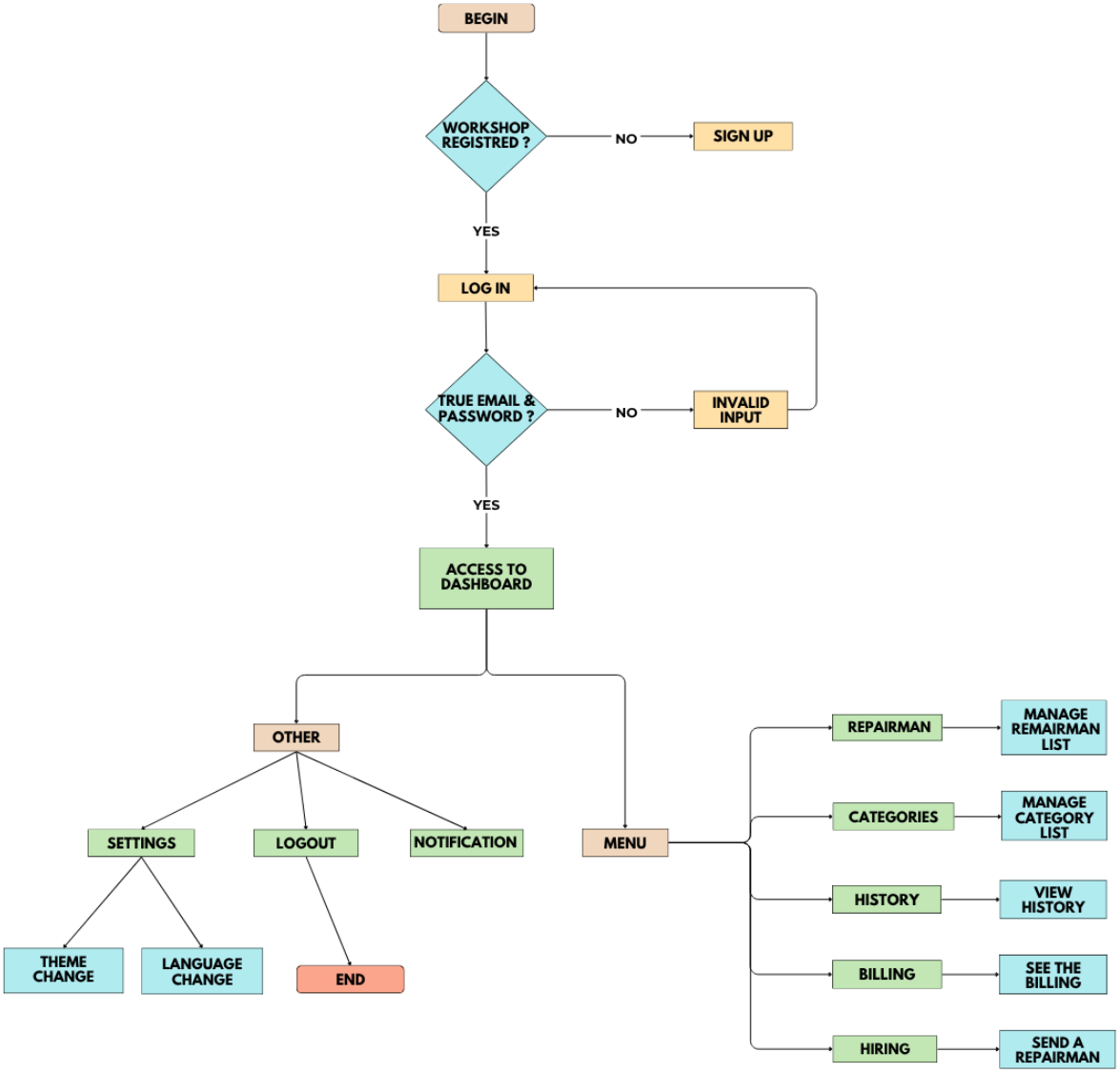

Figure 4.8: User flow diagram for workshop administrators.

4.5.4 Workshop Repairman Flow Diagram

The workshop repairman flow begins with login to access assigned requests, enabling them to update intervention statuses, view their intervention history, and manage their profile for streamlined task execution within the workshop.

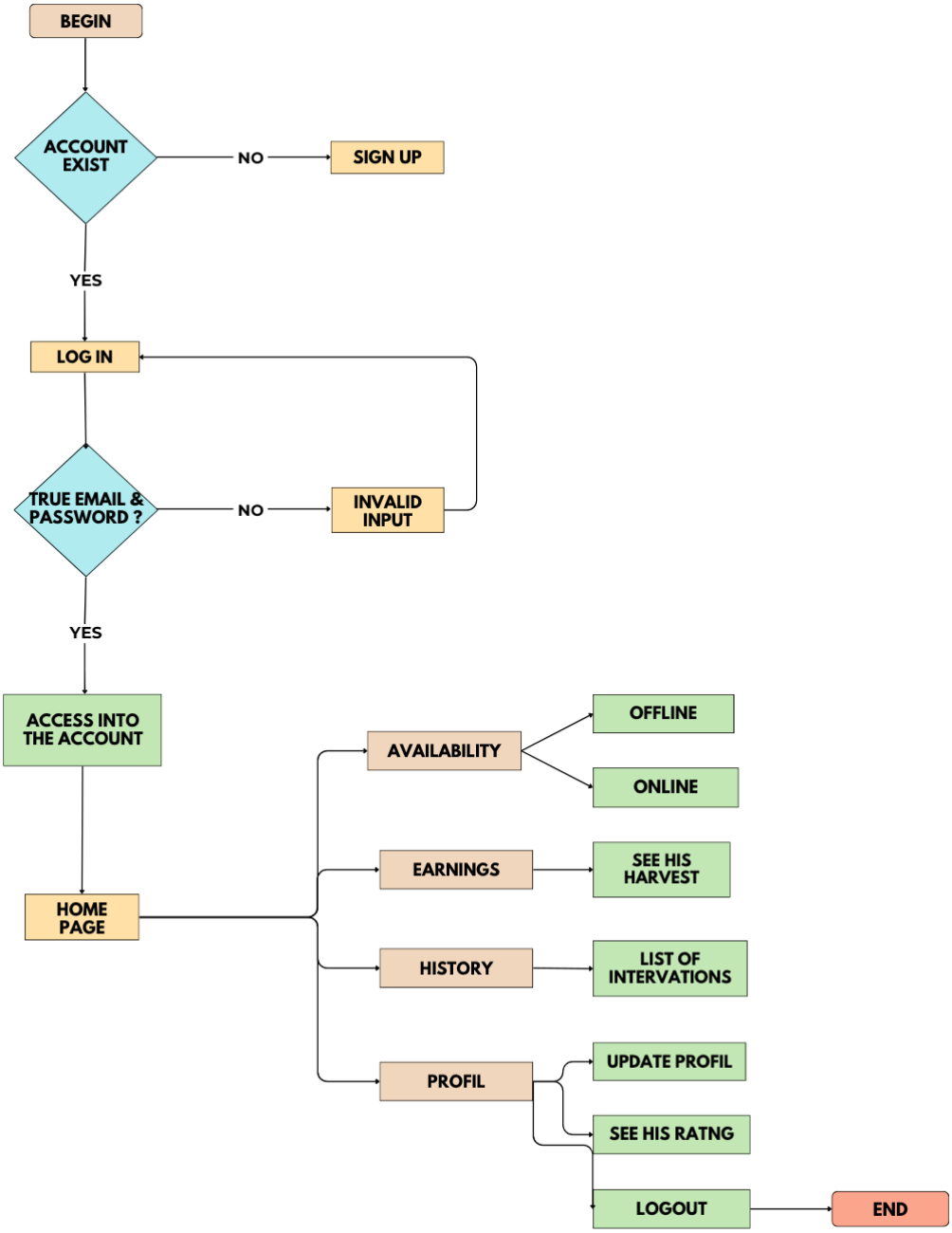

Figure 4.9: User flow diagram for workshop repairmen.

4.5.5 Platform Administrator Flow Diagram

The platform administrator flow starts with login to oversee the system, allowing them to review and approve registrations, manage providers, monitor locations, and generate usage reports to maintain platform-wide efficiency.

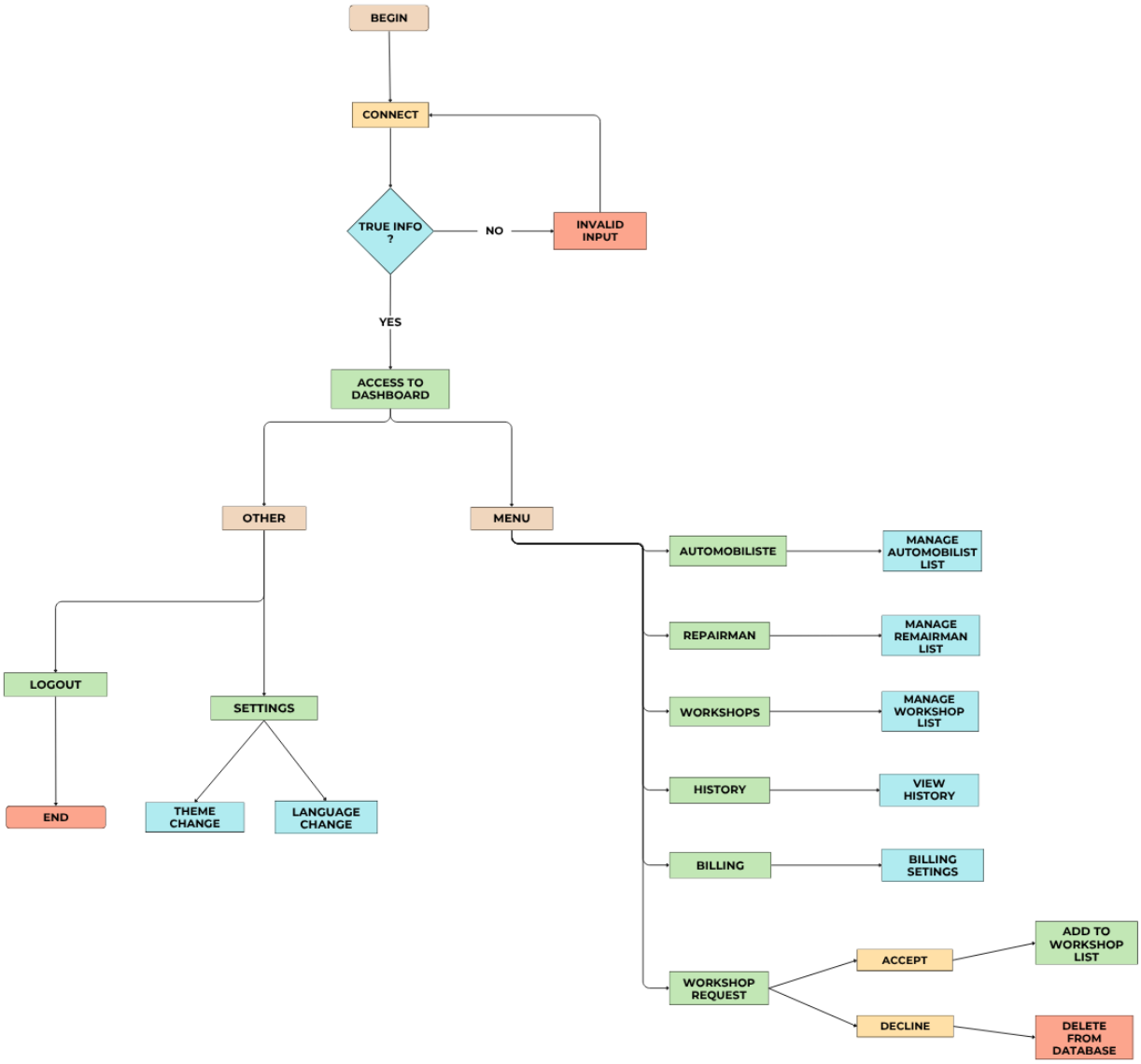

Figure 4.10: User flow diagram for platform administrators.

4.6 Presentation of Our Application

This section presents key interfaces of the application, organized by feature to highlight essential functionalities for each actor.

4.6.1 Home page

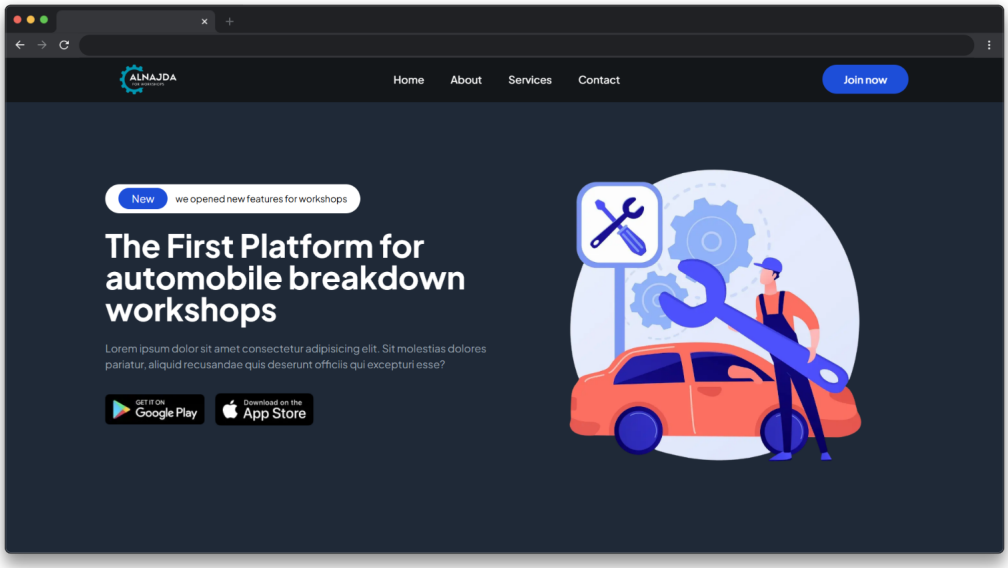

Figure 4.11: Home page.

4.6.2 Registration and Authentication

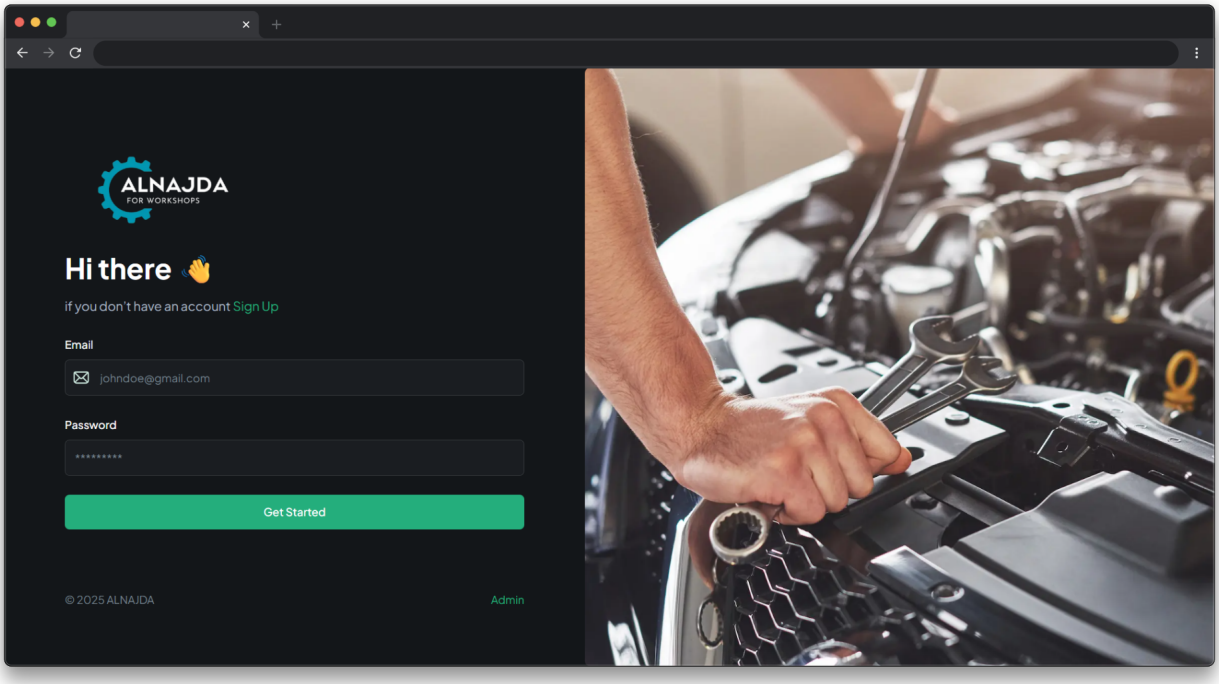

Figure 4.12: Workshop Web login page.

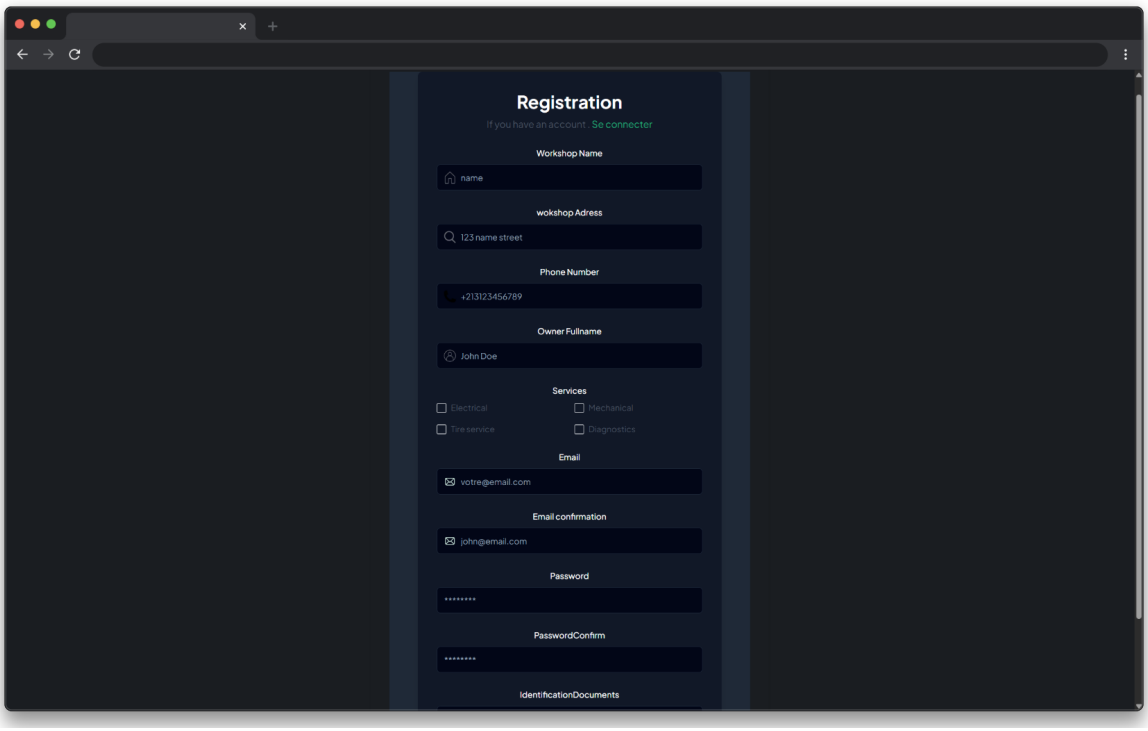

Figure 4.13: Workshop Web registration page.

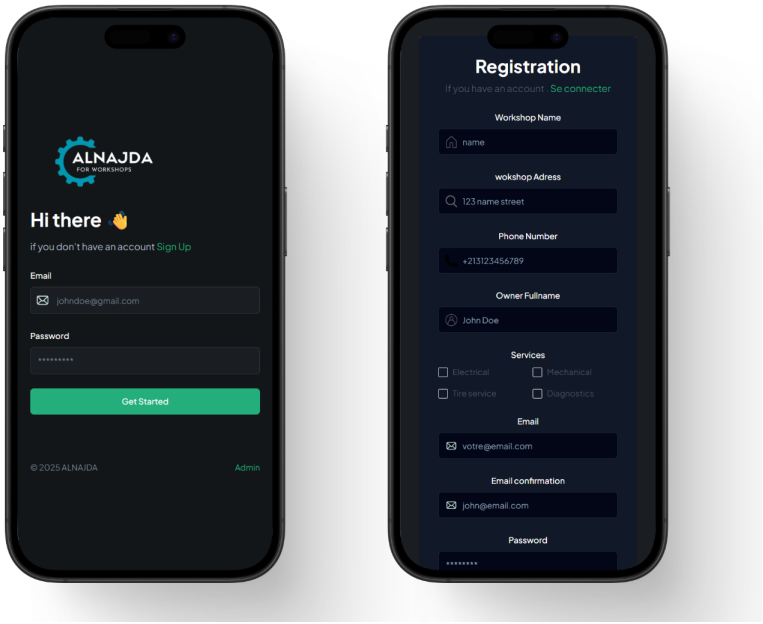

Figure 4.14: Workshop Mobile login and registration pages.

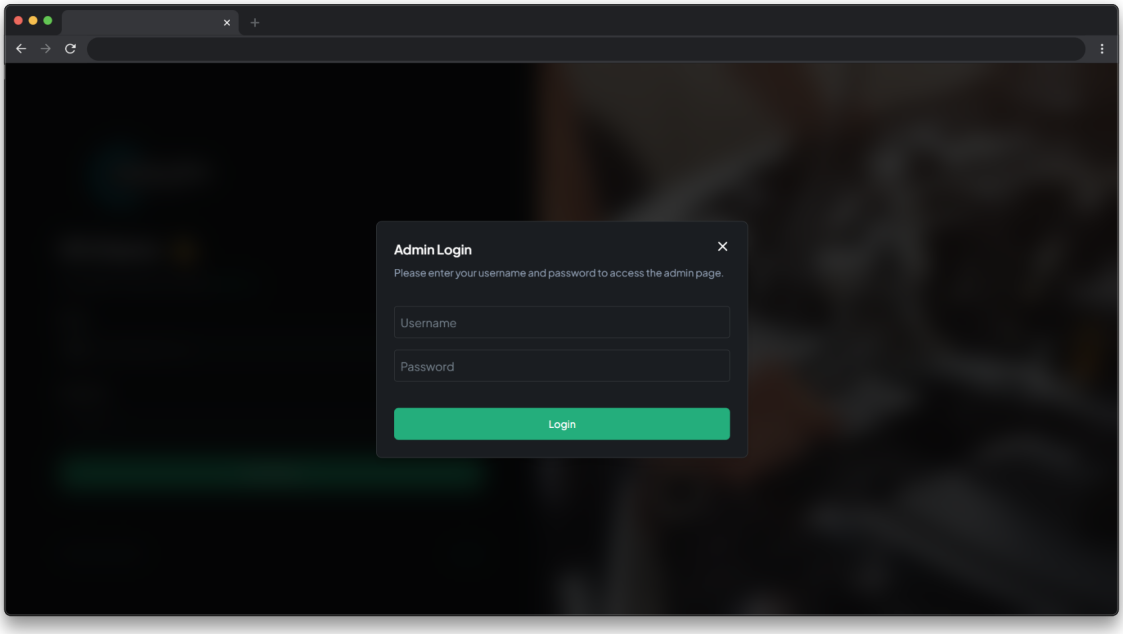

Figure 4.15: Admin login page.

4.6.3 Admin dashboard

4.6.3.1 Admin dashboard home page.

Figure 4.16: Admin dashboard home page.

4.6.3.2 Stranded Motorist list

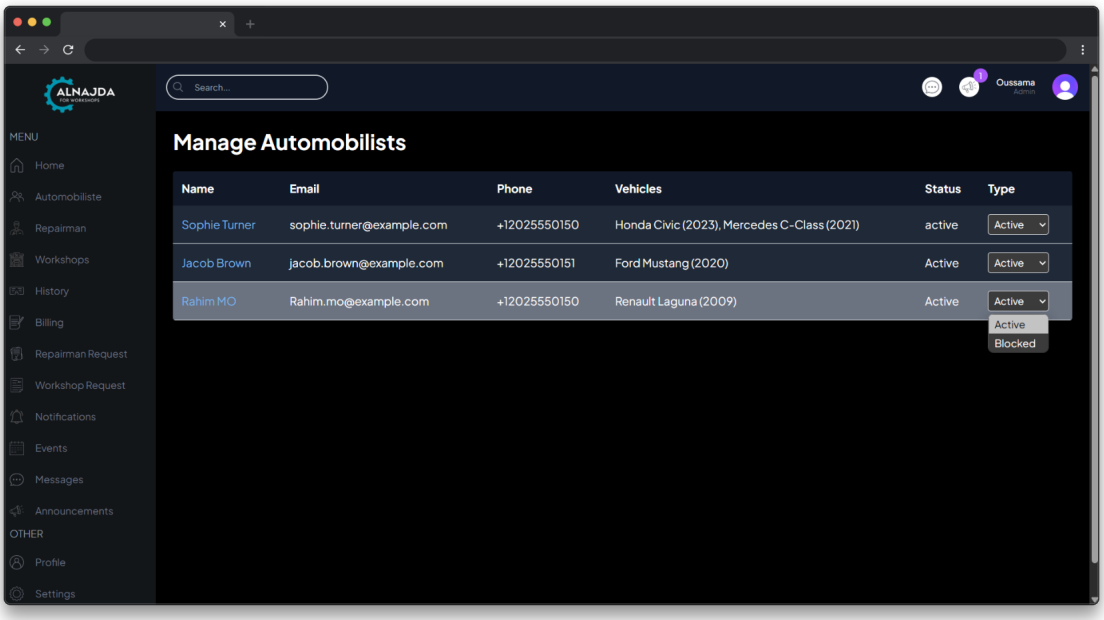

Figure 4.17: Admin dashboard motorists list.

4.6.3.3 Stranded Motorist details

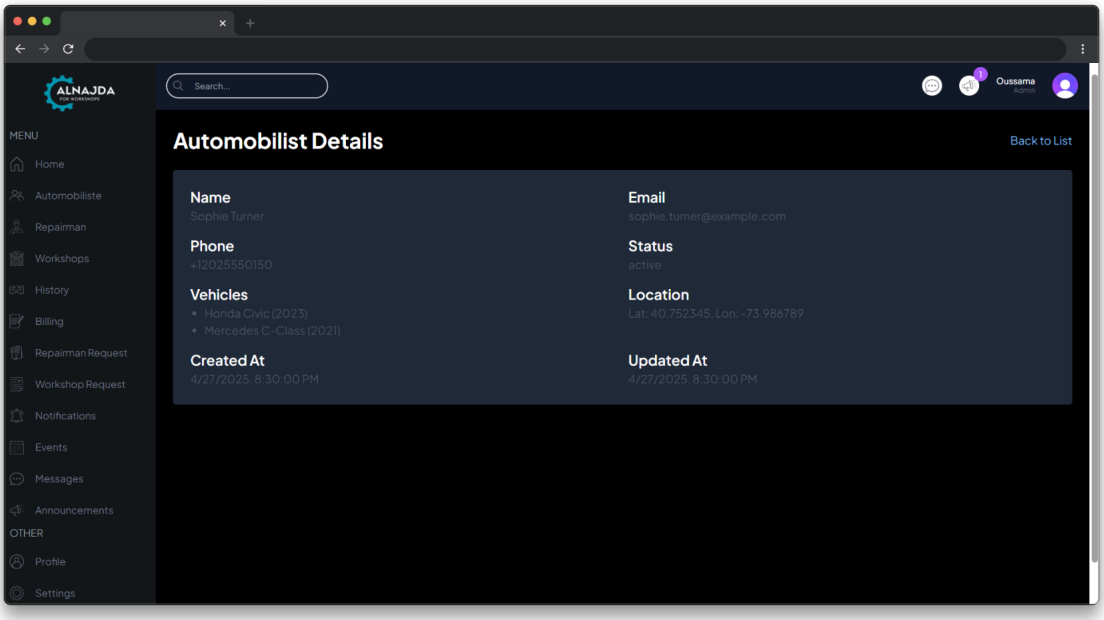

Figure 4.18: Admin dashboard motorist details.

4.6.3.4 Repairman List

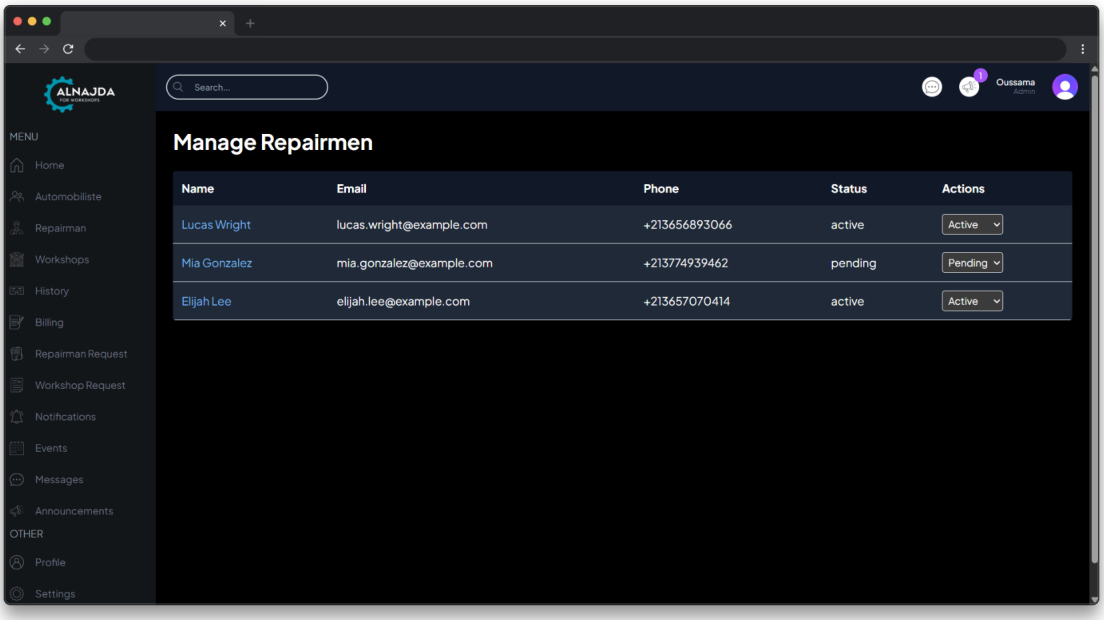

Figure 4.19: Admin dashboard repairman list.

4.6.3.5 Repairman details

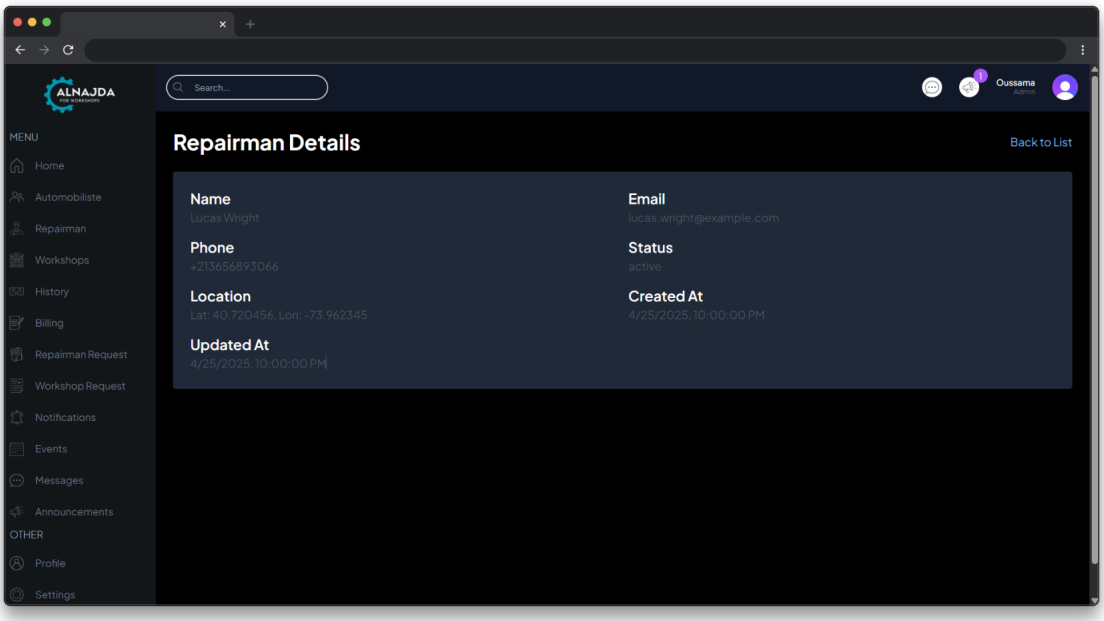

Figure 4.20: Admin dashboard repairman details.

4.6.3.6 Billing Settings

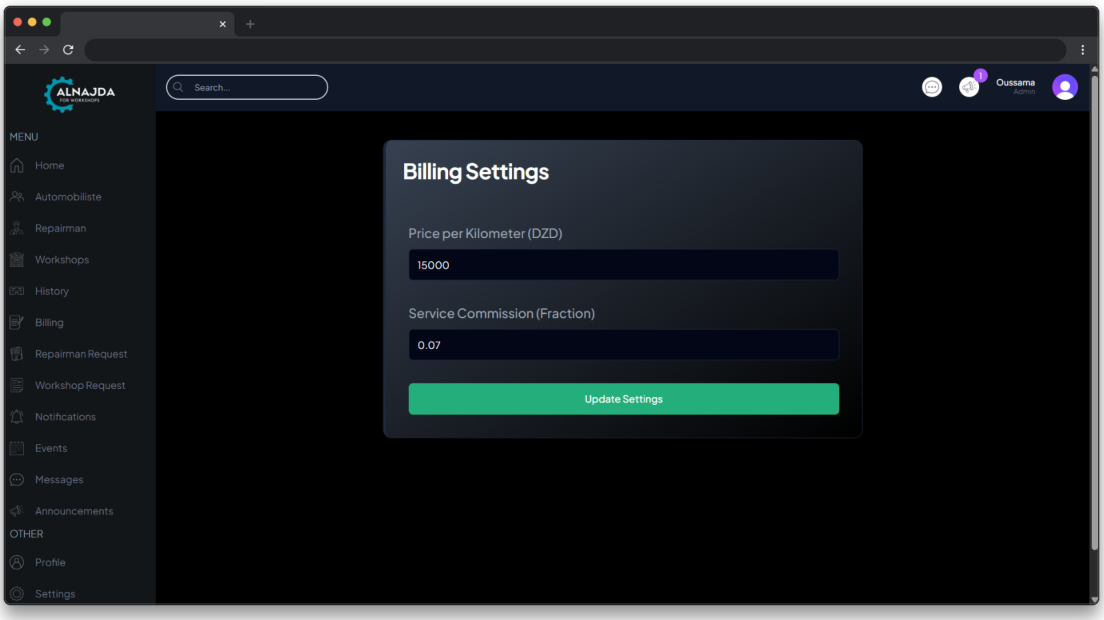

Figure 4.21: Admin dashboard billing settings.

4.6.3.7 Workshop request

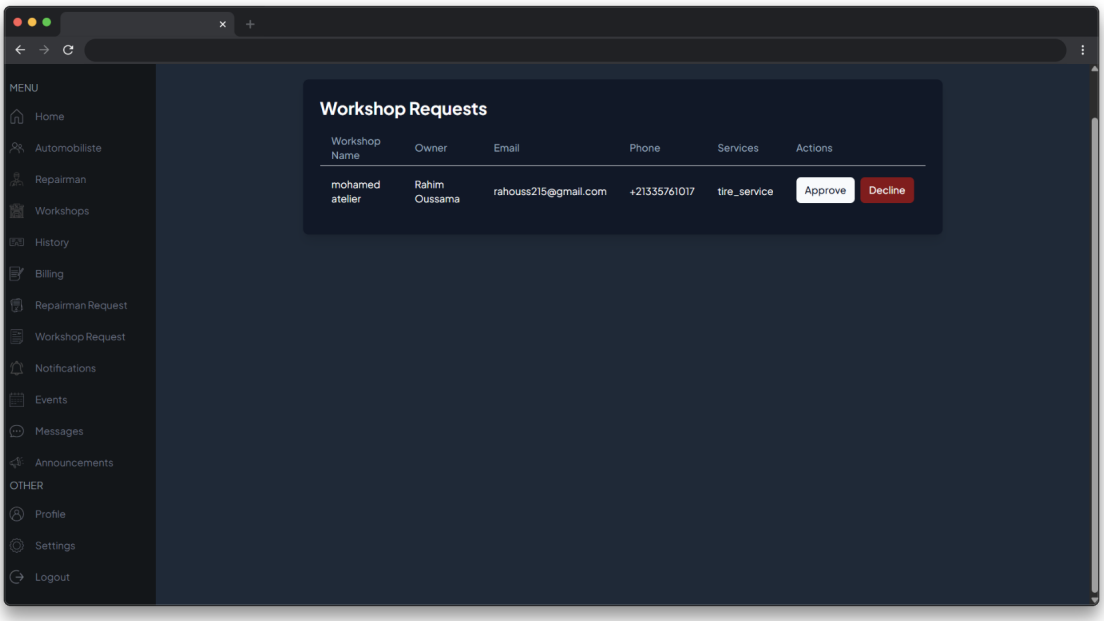

Figure 4.22: Admin dashboard Workshop Request.

4.6.4 Workshop dashboard

4.6.4.1 Repairman List

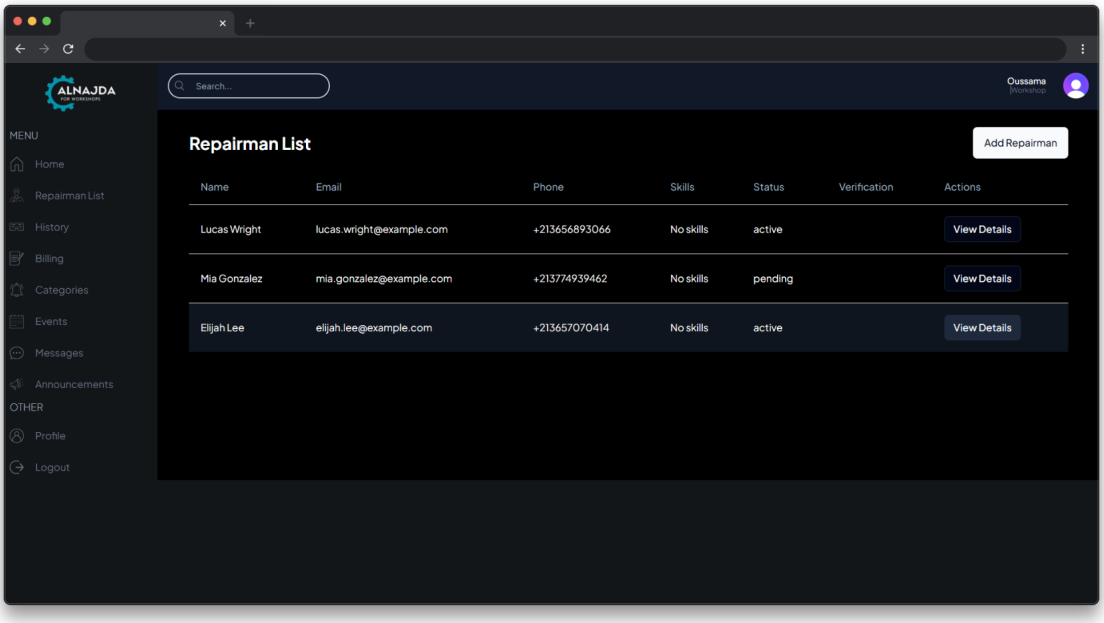

Figure 4.23: Workshop dashboard repairman list.

4.6.4.2 Add a repairman

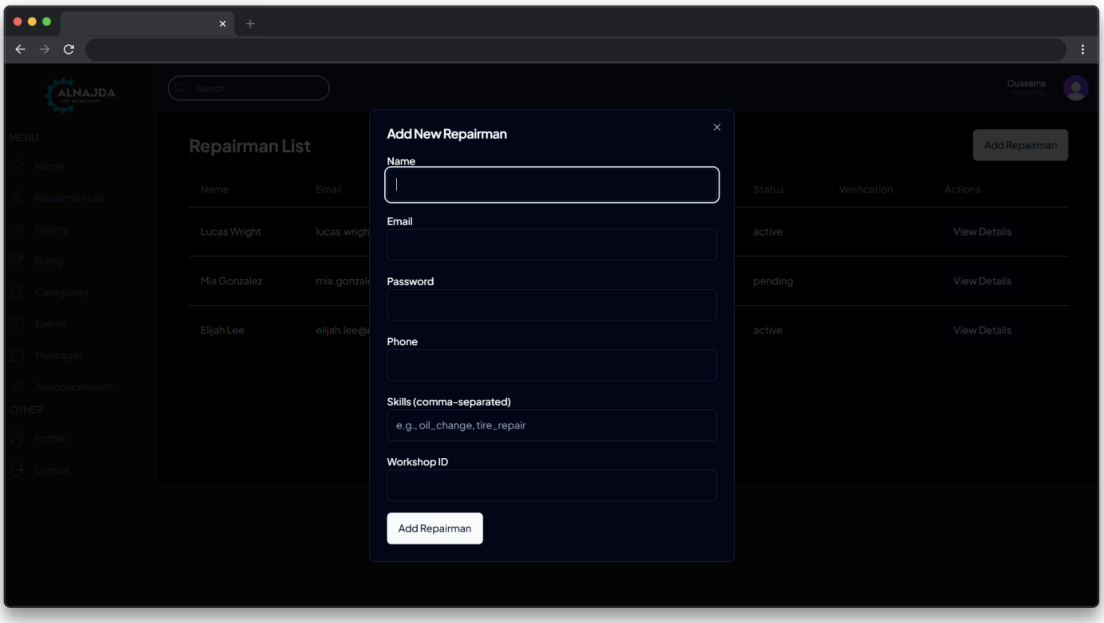

Figure 4.24: Workshop dashboard add a repairman.

4.6.5 Motorist mobile Application

4.6.5.1 Onboarding screens

Figure 4.25: Onboarding mobile app (1/2).

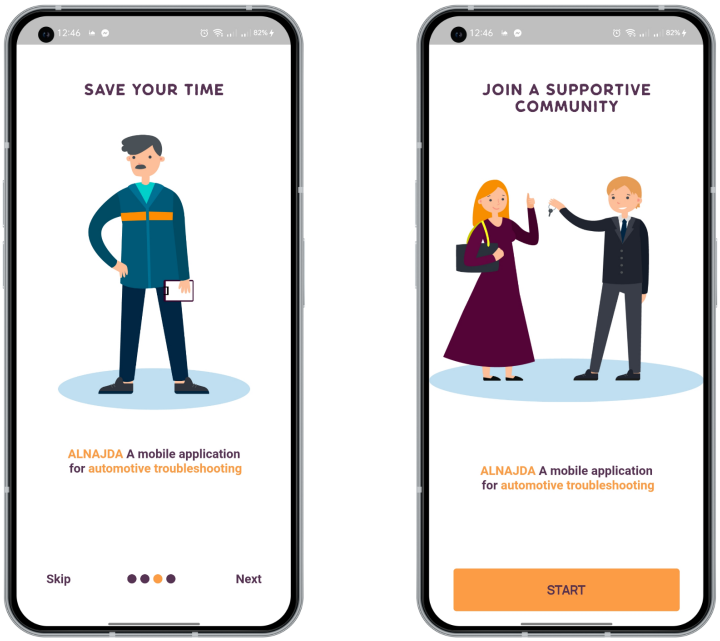

Figure 4.26: Onboarding mobile app (2/2).

4.6.5.2 Login & register screens

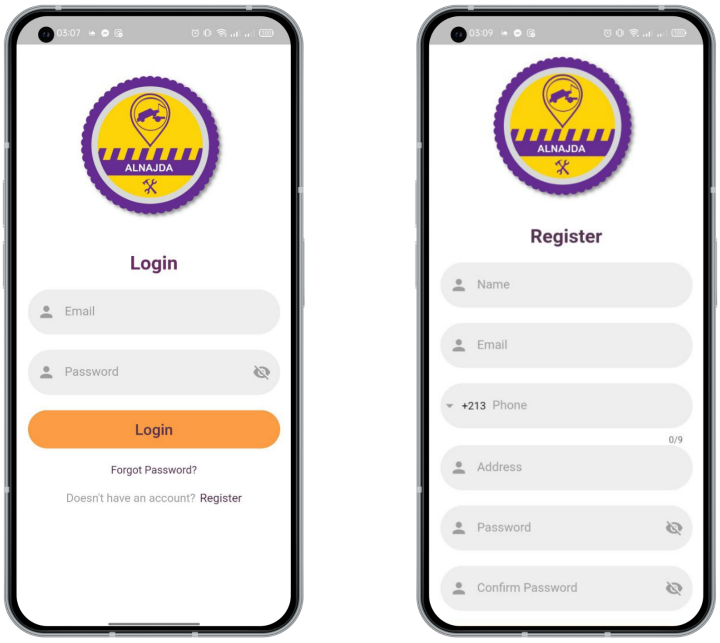

Figure 4.27: Login & register screens.

4.6.5.3 Main screen

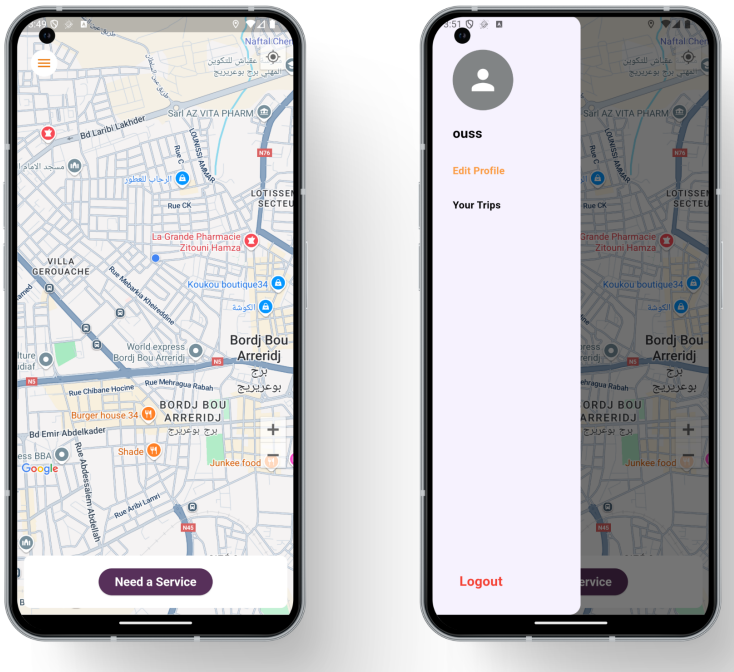

Figure 4.28: Main screen.

4.6.5.4 Request a service screen

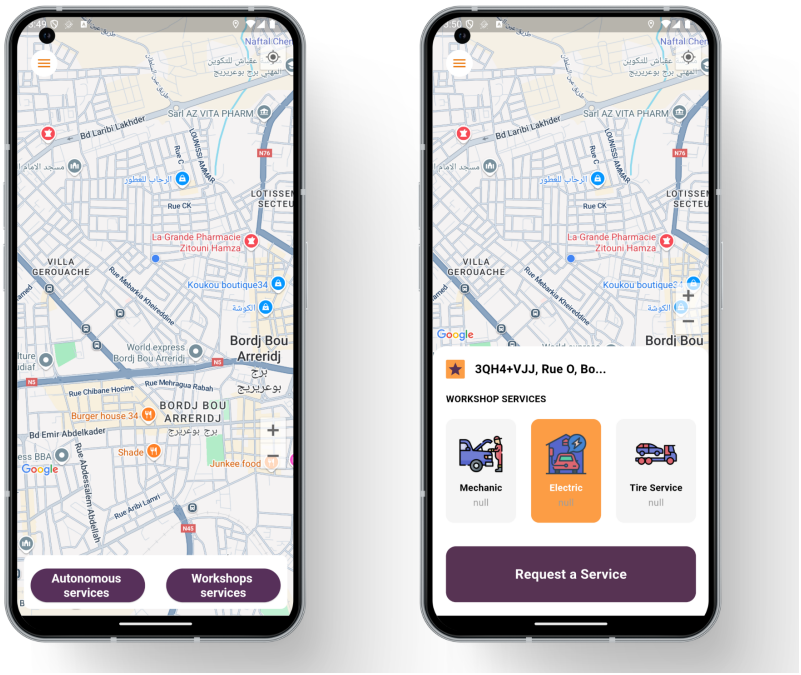

Figure 4.29: Request a service screen.

4.6.5.5 Request updates screen

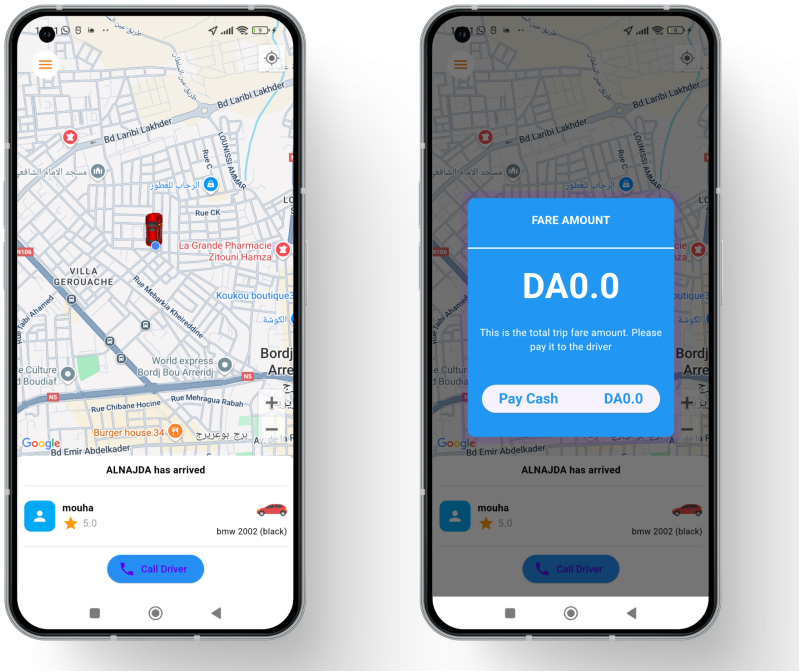

Figure 4.30: Request updates screen.

4.6.5.6 Rating & history screens

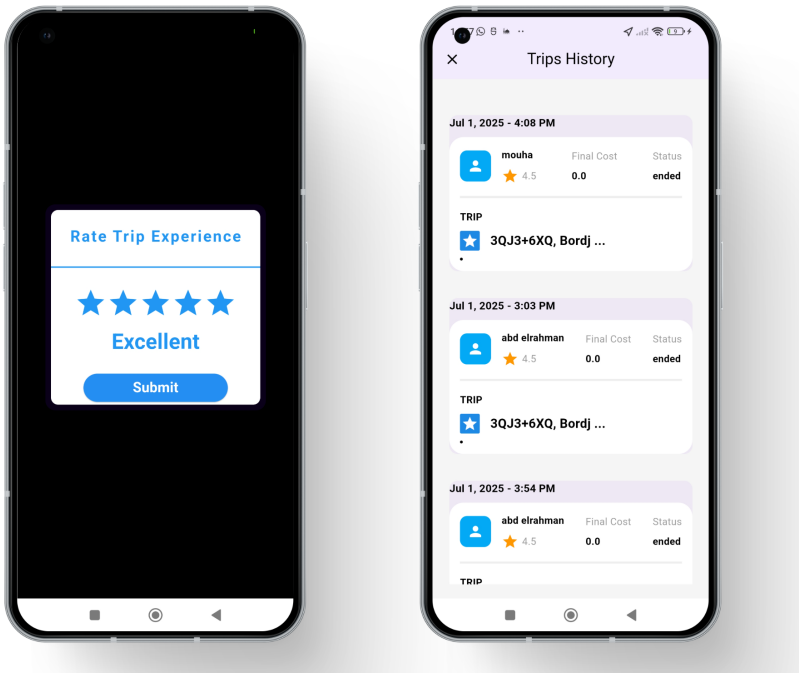

Figure 4.31: Rating & history screens.

4.6.6 Repairmen mobile Application

4.6.6.1 Main screen

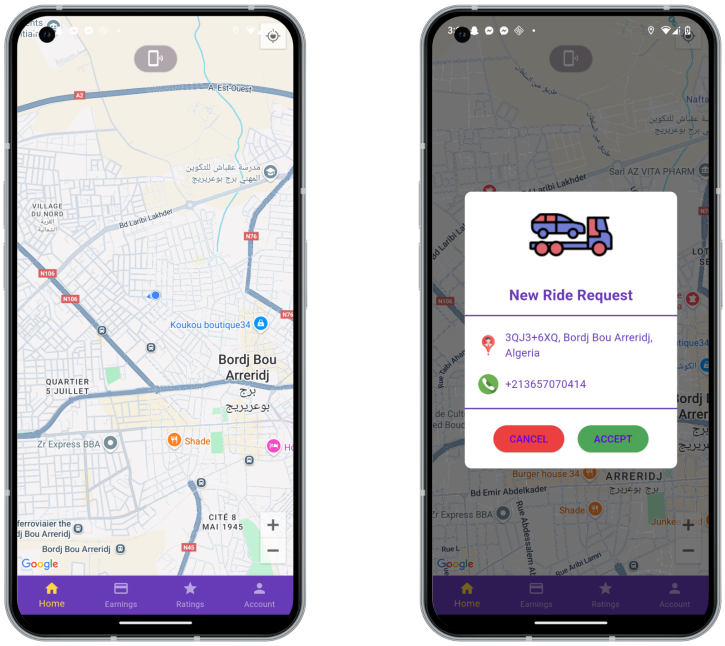

Figure 4.32: Main screen.

4.6.6.2 Request process screen

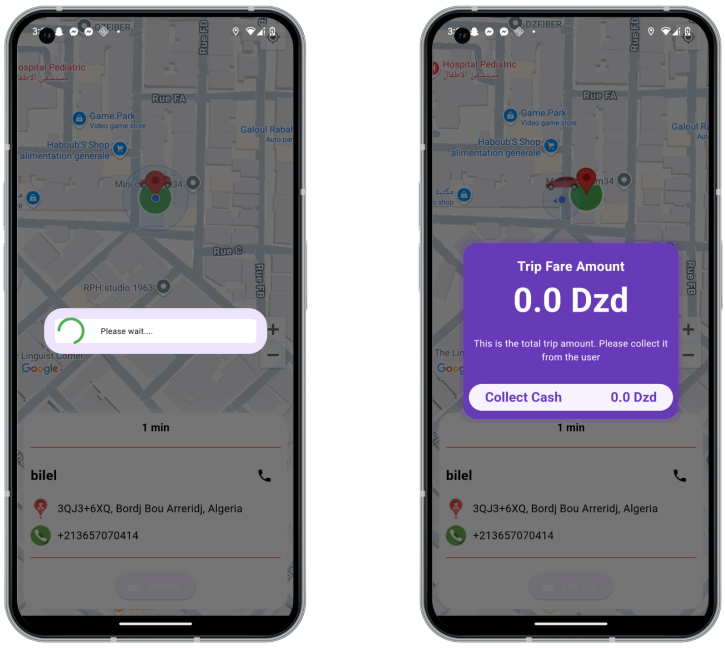

Figure 4.33: Request process screen.

4.6.6.3 Rating & earning screens

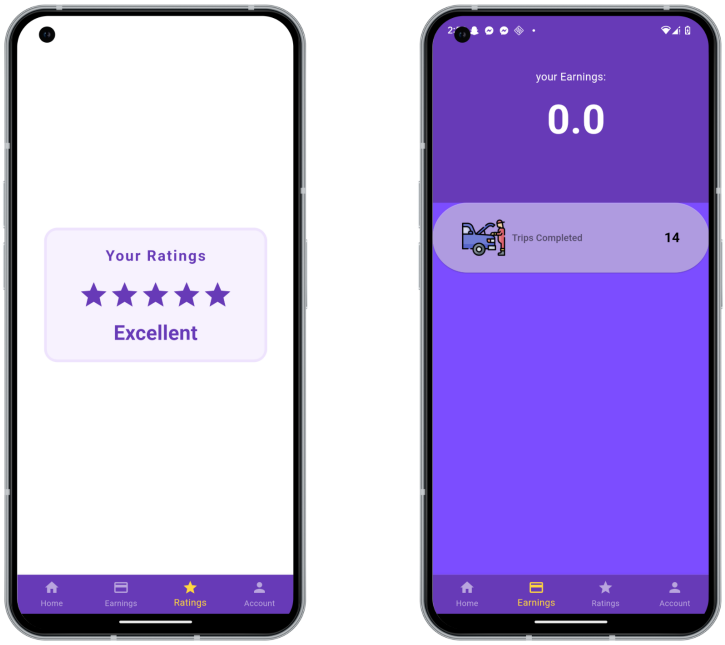

Figure 4.34: Rating & earning screens.

4.7 Conclusion

This chapter detailed the implementation of the vehicle breakdown assistance platform, realizing the design outlined in Chapter 3. The full-stack architecture, leveraging Next.js, Flutter, Node.js, and Firebase, supports scalable and real-time functionalities. The REST API enables seamless interaction between components, addressing user stories like service request creation and payment processing. User flow diagrams and interfaces demonstrate how actors interact with the system, ensuring usability and efficiency. The Agile approach, guided by Scrum events, facilitated iterative development and stakeholder feedback, overcoming challenges like API integration and geospatial query optimization. This implementation establishes a robust, user-centric platform, with insights from testing and validation informing future enhancements, such as eco-friendly technician prioritization.

Chapter 5

General Conclusion

5.1 Contributions

This project developed a robust digital platform that revolutionizes vehicle breakdown assistance in Algeria. The platform integrates a mobile app and web interface, enabling motorists to request and track services in real-time, while providing repairmen and administrators with tools to manage operations efficiently. Key achievements include implementing real-time GPS tracking, secure payment systems, and performance analytics, addressing gaps in existing solutions like DZ Dépannage and Dépani Dz. Using Agile Scrum, the project delivered features such as user registration, service request handling, and intervention tracking across six sprints, enhancing accessibility, reliability, and user satisfaction

5.2 Limitations

The platform has some limitations. It currently lacks a 24/7 chat or helpline feature, which could enhance emergency support. Features such as eco-friendly technician prioritization and partnerships with major workshops were not implemented due to time constraints. Additionally, while designed for scalability with Firebase Realtime Database's automatic scaling and real-time synchronization, the system's performance under high traffic has not been tested, potentially revealing bottlenecks in extreme conditions.

5.3 Future Work

Future enhancements will address these limitations to strengthen the platform. A 24/7 chat or helpline service will be integrated to provide real-time support for motorists. Prioritizing technicians using eco-friendly vehicles will align the platform with sustainability goals. Partnerships with established workshops, such as GME-Performance, will be pursued to expand service coverage. Additionally, stress-testing the platform under high traffic will ensure scalability, positioning it as a leading solution for roadside assistance in Algeria.

References

[1] J. Lonchamp, *Analyse des besoins pour le développement logiciel: Recueil et spécification, démarches itératives et agiles*. Dunod, 2015.

[2] T. Academy, E. Gross, and J. Stanley, *The Project Management Handbook: Simplified Agile, Scrum and DevOps for Beginners*. The Academy, 2021.

[3] Beck, Kent and Beedle, Mike and van Bennekum, Arie and Cockburn, Alistair and Cunningham, Ward and Fowler, Martin and Grenning, James and Highsmith, Jim and Hunt, Andy and Jeffries, Ron and Kern, Jon and Marick, Brian and Martin, Robert C. and Mellor, Steve and Schwaber, Ken and Sutherland, Jeff and Thomas, Dave, "Manifesto for agile software development," 2001, accessed: 2025. [Online]. Available: https://agilemanifesto.org/

[4] M. C. Layton, *Scrum For Dummies*. For Dummies, 2022.

[5] K. Schwaber and J. Sutherland, "The scrum guide: The definitive guide to scrum – the rules of the game," https://scrumguides.org/, 2020, accessed: 2025.

[6] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, *UML @ Classroom: An Introduction to Object-Oriented Modeling*, ser. Undergraduate Topics in Computer Science. Springer, 2015.

[7] G. Harrison, *Next Generation Databases: NoSQL, NewSQL, and Big Data: What every professional needs to know about the future of databases in a world of NoSQL and Big Data*. Apress, 2016.

[8] A. K. S, *Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase*. Packt Publishing, 2018.

[9] Vercel, “Next.js documentation,” https://nextjs.org/docs, accessed: 2025.

[10] Microsoft, “Typescript documentation,” https://www.typescriptlang.org/docs/, accessed: 2025.

[11] Tailwind Labs, “Tailwind css v2 documentation,” https://v2.tailwindcss.com/docs, accessed: 2025.

[12] shadcn, “Shadcn ui documentation,” https://ui.shadcn.com/docs/, accessed: 2025.

[13] Google, “Flutter documentation,” https://docs.flutter.dev/, accessed: 2025.

[14] Node.js Contributors, “Node.js api documentation,” https://nodejs.org/docs/latest/api/, accessed: 2025.

[15] Git Community, “Git documentation,” https://git-scm.com/doc, accessed: 2025.

[16] Postman, Inc., “Postman documentation: Overview,” https://learning.postman.com/docs/introduction/overview/, accessed: 2025.

[17] Microsoft, “Visual studio code documentation,” https://code.visualstudio.com/docs, accessed: 2025.

[18] Google, “Google maps platform documentation,” https://developers.google.com/maps/documentation, accessed: 2025.

[19] NextAuth.js Contributors, “Nextauth.js documentation,” https://next-auth.js.org/, accessed: 2025.