People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University Mohamed El Bachir El Ibrahimi — BBA
Faculty of Sciences and Technology
National Institute of Electronics and Telecommunications
Department of Electronics — Industrial Electronics Specialty

A IoT-Enabled Electric Scooter Rental System with Docking Stations

RAFEEQ

Prepared by:

Mohamed Bouhalli · Mohamed Benrekia · Yassine Cherrat

Encadrant: Dr. Seif Eddine AZOUG
Maitre Conférences B

President: Dr. Idriss MESSAOUDENE
Maitre conférences A

Examinateur: Dr. Tarek ABED
Maitre Conférences B

Représentante Incubateur :Dr. Hadjer SINACER
Maitre Conférences B

Academic year 2025/2026

Dedication

Mohamed Bouhalli

Alhamdulillah

Thank you Mom,
Thank you Dad,

Thank you to my family Thank you Aymen Thank you
Roumaissa Thank you Abdou Thank you Abdollah
Thank you baraa Thank you Auntie

Thank you, Mohamed. Thank you, Yassine.

Thank you mahdi Thank you Rahim Thank you fofo

Thank you ayoub Thank you Younes Thank you djaber Thank you Abdelkarim

and Thank you to all my MCIL classmates

, thank you to my teachers,
thank you to my colleagues.

Thank you to everyone who contributed,
even with just a sincere word.

— MOHAMED BOUHALLI

Dedication

Mohamed Benrekia

To my beloved parents, *with the deepest expressions of gratitude and appreciation for their endless support, great patience, and sincere prayers that have been the light guiding my path. You are the reason behind every success I have achieved, and without you, this work would not have been possible. May God bless you and keep you always.*

To my late aunt, *who was a true example of kindness and tenderness. May God have mercy on her and grant her a place in His vast paradise. Her beautiful memory will always remain a light accompanying me throughout my journey.*

To my dear grandparents, *a source of blessing and wisdom, whose love and heartfelt prayers have been a constant support throughout my journey.*

To my dear family, near and far, *for their sincere love, continuous encouragement, and unwavering belief in me, which gave me the strength to move forward.*

To my brothers and sisters, *companions of the journey and pillars of support, who have always stood by my side through every moment.*

To my thesis friends, Anes and Yassine, *who shared this journey with me and were a true source of support.*

To my study friends: *Rahim, Mehdi, Ayoub, Farouk, Younes, Djaber, and Abdelkarim and to all my MCIL classmates*

who accompanied me throughout my academic journey and contributed to the success of this work. Special thanks to Ayoub (the taxi driver) for his valuable help in transporting equipment and materials.

To my teachers and mentors, *for their valuable guidance and continuous support, which played a significant role in the completion of this work.*

To every Algerian citizen *who faces daily transportation challenges, I dedicate this work, hoping to contribute, even in a small way, to improving everyday life.*

— MOHAMED BENREKIA

Dedication

Yassine Cherrat

"Whoever says 'I am worthy of it' attains it, and whoever walks steadily toward their dream reaches their goal."

All praise is due to Allah, by whose grace all good deeds are completed, and who has enabled me to accomplish this work after years of diligence and perseverance. To Him alone belongs all praise, in the beginning and in the end.

I dedicate the fruit of this effort to my noble parents, who have always been my greatest support and strength, and who instilled in me the love of knowledge and the determination to succeed. No matter how many words I write, I can never fully repay them.

***To my beloved mother,** who accompanied me with her prayers and tenderness throughout all stages of my life, whose smile was a source of comfort and strength in the most difficult moments.*

***To my dear father,** who wore out his effort and patience for me, striving with all he had so that I may reach this level, and who has been my support and role model on the path of success.*

***To my two grandmothers,** who surrounded me with love and sincere prayers, and whose presence has brought beautiful meaning into my life. I ask Allah to protect them and grant them continued health and well-being.*

***To my dear brothers and sisters,** who shared with me moments of hardship before joy, and stood by my side throughout the different stages of this journey.*

***To my respected teachers,** who illuminated my path of knowledge and provided me with education and guidance, playing a major role in my academic formation.*

***To my friends and colleagues,** who accompanied me throughout my studies, sharing both the challenges and the beautiful memories of this journey.*

And to everyone who offered me help, encouraged me with kind words, or believed in my ability to reach this moment.

*Finally, I dedicate this achievement **to myself**, who remained patient, hardworking, and resilient despite all difficulties, making this success the fruit of faith in Allah and continuous effort.*

— YASSINE CHERRAT

Acknowledgements

In the name of Allah, the Most Gracious, the Most Merciful. All praise is due to Him for granting us the strength, the patience, and the guidance necessary to bring this work to completion.

We wish to express our sincerest gratitude to our supervisor, **Dr. Saif Eddine Azzoug**, for accepting to direct this thesis, for the trust he placed in our project from its earliest stages, and for the rigorous attention with which he reviewed our work along the way. His insightful remarks and his constant availability have substantially shaped the final form of this document.

We extend our gratitude to the President and members of the jury for the honour they grant us by evaluating this work, and for the time they have devoted to reading and discussing our manuscript.

Our acknowledgements go to the entire faculty of the **National Institute of Electronics and Telecommunications** at the University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj. Throughout the five years of our cycle, the teachers and the administrative staff of the Institute have provided us with the knowledge, the practical skills, and the institutional support without which this project would not have been possible.

A special word of thanks goes to the technical staff of the electronics and mechanical workshops, who patiently assisted us through countless fabrication, soldering, and 3D-printing sessions, and who often shared their experience well beyond what was required of them.

We thank our friends and colleagues of the Industrial Electronics specialty, with whom we shared the long hours of study and the equally long hours of debugging. The intellectual exchange and the camaraderie of our cohort have been one of the greatest privileges of these years.

Finally, we owe our deepest debt to our families. To our parents, who supported us in every possible way and never ceased to encourage us; to our brothers and sisters, who endured our absences and our preoccupations during the most intense periods of the work; to all our relatives whose prayers and confidence in us never wavered. This work is, in a very real sense, theirs as much as ours.

Mohamed Bouhalli
Mohamed Benrekia
Yassine Cherrat

Bordj Bou Arreridj, June 2026

Abstract

Urban mobility in Algerian cities suffers from a structural void in the one-to-three-kilometre trip range: public buses are too rigid, taxis too expensive, private cars too inefficient, and walking too constrained by climate and infrastructure. This thesis presents the design, implementation, and validation of **Rafeeq**, a smart electric scooter rental platform conceived to address this First and Last Mile problem in the Algerian institutional, economic, and cultural context.

The platform integrates four interoperable subsystems. A connected fleet of Xiaomi M365 electric scooters is instrumented with a Raspberry Pi 3 Model B+ onboard computer paired with a SIM800L 2G GPRS modem for cellular uplink, and communicates with the scooter's electronic speed controller through the Bluetooth Low Energy Nordic UART service. A network of physical docking stations equipped with ESP32 microcontrollers drives 12 V electromechanical Solenoid locks through TIP122 Darlington power stages, with full electrical protection including flyback diodes and LM7805 regulated rails. A cloud backend implemented in FastAPI and deployed on Render exposes thirty-two REST endpoints and maintains real-time WebSocket gateways with both the IoT endpoints and the mobile clients; user data is persisted in a PostgreSQL 16 database with strict ACID guarantees on all wallet operations. A cross-platform Flutter mobile application covers sixteen user screens across the full rental journey, with full right-to-left Arabic localisation, French, and English support.

The system was validated through a six-family experimental campaign covering electronic bench tests, BLE communication, mechanical lock characterisation, end-to-end latency measurement, backend load tests, and mobile user trials. The prototype achieves an end-to-end unlock latency of 2.83 seconds against a 3-second target, a Solenoid holding force of 73 N against a 50-N target, an onboarding completion time of 78 seconds against a 90-second target, and a station continuous power consumption of 84 W against a 120-W ceiling. The validation demonstrates that an integrated, locally developed smart-mobility platform can technically address the First and Last Mile problem in the Algerian context, and positions Rafeeq as a credible candidate for a subsequent pilot deployment at the University Mohamed El Bachir El Ibrahimi campus.

Keywords: Smart mobility, electric scooter, IoT, Raspberry Pi, ESP32, BLE, FastAPI, Flutter, First and Last Mile, Algeria, docking station, urban transportation.

Résumé

La mobilité urbaine dans les villes algériennes souffre d'un vide structurel sur les trajets de un à trois kilomètres : les bus sont trop rigides, les taxis trop chers, les voitures personnelles trop inéfficaces, et la marche trop contrainte par le climat et les infrastructures. Cette mémoire présente la conception, la réalisation et la validation de **Rafeeq**, une plate-forme intelligente de location de trottinettes électriques conçue pour répondre à ce problème du *premier et dernier kilomètre* dans le contexte institutionnel, économique et culturel algérien.

La plate-forme intègre quatre sous-systèmes interopérables : une flotte de trottinettes Xiaomi M365 instrumentées d'un Raspberry Pi 3 Modèle B+ embarqué, couplé à un modem SIM800L 2G GPRS pour la liaison cellulaire montante, et communiquant avec le contrôleur ESC du scooter via le service Nordic UART de Bluetooth Low Energy; un réseau de stations d'accueil physiques pilotées par des microcontrôleurs ESP32 qui commandent des serrures électromécaniques 12 V à travers un étage de puissance TIP122; un backend cloud FastAPI déployé sur Render exposant trente-deux *endpoints* REST et des passerelles WebSocket temps réel; et une application mobile Flutter multi-plate-forme couvrant seize écrans utilisateur avec support intégral de l'arabe (droite à gauche), du français et de l'anglais.

Le système a été validé par une campagne expérimentale en six familles de tests. Le prototype atteint une latence de déverrouillage de bout en bout de 2,83 s pour une cible de 3 s, une force de maintien de la serrure de 73 N pour une cible de 50 N, un temps d'inscription de 78 s pour une cible de 90 s, et une consommation continue de la station de 84 W pour un plafond de 120 W. La validation démontre qu'une plate-forme de mobilité intelligente développée localement peut techniquement répondre au problème du premier et dernier kilomètre dans le contexte algérien.

Mots-clés : Mobilité intelligente, trottinette électrique, IoT, Raspberry Pi, ESP32, BLE, FastAPI, Flutter, premier et dernier kilomètre, Algérie, station d'accueil, transport urbain.

ملخص

تعاني التنقلات الحضرية في المدن الجزائرية من فراغ هيكلي في الرحلات التي تتراوح بين كيلومتر واحد وثلاثة كيلومترات؛ فالحافلات العمومية بطيئة جدًا، وسيارات الأجرة مكلفة، والسيارات الخاصة غير فعّالة، كما أن المشي مقيّد بالظروف المناخية والبنية التحتية. تعرض هذه الأطروحة تصميم وتنفيذ والتحقق من نظامRafeeq ، وهو منصة ذكية لتأجير الدراجات الكهربائية، تم تطويرها لمعالجة مشكلة "الميل الأول والميل الأخير "ضمن السياق المؤسسي والاقتصادي والثقافي الجزائري.

تدمج المنصة أربعة أنظمة فرعية مترابطة. حيث يتم تجهيز أسطول متصل من دراجات Xiaomi M365 الكهربائية بحاسوب مدمج من نوع Raspberry Pi 3 Model B+ مقترن بمودم SIM800L 2G GPRS لربط الشبكة الخلوية، ويتواصل مع وحدة التحكم في سرعة الدراجة عبر واجهة UART. Bluetooth Low Energy (BLE) كما تعتمد شبكة من محطات التثبيت (Docking Stations) المزودة بوحدات تحكم ESP32 على تشغيل أقفال كهروميكانيكية بجهد 12 فولت من نوع Solenoid باستخدام مراحل قدرة TIP122 Darlington، مع حماية كهربائية كاملة تشمل دايودات الحماية (flyback) ومنظمات الجهد.LM7805

تم تنفيذ الواجهة الخلفية السحابية باستخدام FastAPI ونشرها علىRender ، حيث توفر ثلاثين نقطة REST وتحافظ على بوابات WebSocketفي الزمن الحقيقي لكل من أجهزة إنترنت الأشياء والتطبيقات المحمولة. يتم تخزين بيانات المستخدمين في قاعدة بيانات PostgreSQL 16 مع ضمانات ACID صارمة لجميع عمليات المحفظة. كما يغطي تطبيق الهاتف المحمول متعدد المنصات المبني بـ Flutter ستة عشر شاشة مستخدم تغطي كامل رحلة التأجير، مع دعم كامل للغة العربية (من اليمين إلى اليسار)، بالإضافة إلى الفرنسية والإنجليزية.

تم التحقق من النظام من خلال حملة تجريبية تشمل ستة محاور: اختبارات إلكترونية، اتصالBLE ، توصيف القفل الميكانيكي، قياس زمن التأخير من طرف إلى طرف، اختبارات التحمل في الخلفية، واختبارات واجهة الهاتف. أظهر النموذج الأولي زمن فتح/غلق قدره 2.83 ثانية مقابل هدف 3 ثوانٍ، وقوة تثبيت للقفل 73 نيوتن مقابل هدف 50 نيوتن، وزمن تسجيل مستخدم جديد 78 ثانية مقابل هدف 90 ثانية، واستهلاك طاقة مستمر للمحطة قدره 84 واط مقابل حد أقصى 120 واط.

تُظهر هذه النتائج أن منصة تنقل ذكية متكاملة ومطورة محليًا يمكنها معالجة مشكلة "الميل الأول والميل الأخير "تقنيًا في السياق الجزائري، وتضع نظام Rafeeq كمرشح موثوق لإطلاق تجربة ميدانية لاحقة في جامعة محمد البشير الإبراهيمي.

الكلمات المفتاحية:
التنقل الذكي، الدراجة الكهربائية، إنترنت الأشياء، Raspberry Pi، ESP32، BLE، FastAPI، Flutter، الميل الأول والأخير، الجزائر، محطات التثبيت، النقل الحضري

Contents

List of Figures

List of Tables

General Introduction

Urban mobility has become one of the most consequential challenges facing Algerian cities in the present decade. As the country urbanises and its vehicle fleet expands at a rate that vastly exceeds the growth of its road infrastructure, citizens are confronted every day with congested arteries, unreliable public transport, overpriced taxis, and the slow attrition of walking through inadequate sidewalks and a difficult climate. These pressures fall most heavily on the short trips of one to three kilometres that constitute the bulk of urban movement: the trip to the campus from the residence hall, the trip from the bus terminal to the office, the trip from the parking lot to the shopping centre. This is the so-called **First and Last Mile** problem, and no existing mode of transport in Algeria addresses it well.

The present thesis grew out of a simple observation. As students at the University Mohamed El Bachir El Ibrahimi in Bordj Bou Arreridj, the authors of this work experience the First and Last Mile gap at first hand every day. The 1.8-kilometre distance between the main university residence and the central campus is too long to walk comfortably in summer, too short to be served well by the limited ONOU shuttle fleet, and too inconvenient to be served at all by taxis. Multiplied across eighteen thousand students, this single operational void produces tens of thousands of cumulative hours of lost time every academic year. A direct, technical, and locally developed answer to this problem is both possible and overdue.

That answer is **Rafeeq**: a smart electric scooter rental platform designed from the ground up for the Algerian institutional, economic, and cultural context. Rafeeq integrates four interoperable subsystems — a connected fleet of Xiaomi M365 electric scooters instrumented with embedded Raspberry Pi computers, a network of physical docking stations driven by ESP32 microcontrollers, a cloud backend implemented in FastAPI on Render, and a cross-platform Flutter mobile application — to deliver an end-to-end rental experience in which a user opens an application, walks to a nearby station, scans a QR code, and rides away. At the end of the trip, the user returns the scooter to any station in the network, the lock engages automatically, and the platform debits the wallet. The entire interaction is unattended, secure, and designed to take less time than the alternative of waiting for a bus.

The work presented in this thesis covers the complete engineering cycle of the Rafeeq platform, from the identification of the problem through the design, fabrication, and validation of a working prototype. The manuscript is organised in three chapters.

Chapter 1 establishes the problem context. It opens with an overview of the Algerian transportation system, surveys the limitations of buses, private cars, taxis, and walking for

short urban trips, develops the urban congestion problem in quantitative terms, formalises the First and Last Mile gap in the Algerian and the campus context, presents the results of a primary survey carried out with the target user population, and concludes by introducing Rafeeq as the proposed solution and stating the project objectives.

Chapter 2 addresses the state of the art and the conceptual foundations on which Rafeeq is built. It reviews the existing classes of smart mobility platforms operating internationally, characterises their business models and technical architectures, identifies the technological building blocks transferable to the Algerian context, and positions Rafeeq within this landscape. The chapter closes with the architectural decisions that translate the abstract requirements of the previous chapter into a concrete technical specification.

Chapter 3 presents the design, implementation, and validation of the working prototype. It describes the electronic architecture of the docking station, the IoT subsystem embedded in each scooter, the backend services and the database schema, the mobile application, the end-to-end communication protocols, and the security architecture. It then reports the experimental validation campaign organised in six families of tests — electronic bench tests, BLE communication, mechanical lock characterisation, end-to-end latency measurement, backend load tests, and mobile user trials — and discusses the results obtained against the pre-defined performance targets.

The manuscript closes with a general conclusion that summarises the contributions of the work, situates them against the stated objectives, and outlines the perspectives for the subsequent pilot deployment at the BBA campus.

CHAPTER 1

Transportation Problem in Algeria and the Rafeeq Solution

1.1 Introduction

Urban mobility is one of the founding pillars of contemporary social and economic life. The ease, the speed, and the affordability with which a citizen can move within a city shape, in a very direct way, their access to employment, education, healthcare, and social participa-tion. In Algeria, the rapid demographic expansion of cities over the last three decades, the structural multiplication of the private vehicle fleet, and the operational limits of existing public transport networks have combined to produce a chronic mobility crisis that is now experienced daily by millions of citizens [1, 2, 3].

This crisis is not uniform across modes or across territories. It is particularly acute in dense urban centres such as Algiers, Oran, Constantine, and Annaba, where peak-hour congestion has reached levels comparable to those observed in European metropolitan areas of similar size. It is equally acute, though in a different form, in mid-sized university cities such as Bordj Bou Arreridj (BBA), Sétif, Tizi-Ouzou, and Constantine, where students must perform short daily trips between residences and academic facilities — distances of one to three kilometres that fall in an awkward range: too long to be walked comfortably, too short to be efficiently served by mass-transit systems, and too expensive to be served by taxis on a routine basis [4, 5].

This chapter establishes the contextual and analytical foundation upon which the *Rafeeq* project is built. Section 1.2 presents the institutional and operational organisation of the Algerian transport system, recalling its main actors and the strategic role of urban mobility in national development. Section 1.3 examines, mode by mode, the structural deficiencies that affect daily commuters: collective buses, private vehicles, taxis, and walking. Section 1.4 analyses urban congestion as a systemic phenomenon and quantifies its consequences on time, public health, and the urban economy. Section 1.5 introduces the well-documented **First and Last Mile** problem and demonstrates how it manifests in the specific context of Algerian university campuses. Section 1.6 evaluates the reasons why none of the existing modes can adequately answer this problem. Finally, Section 1.7 introduces *Rafeeq* — a smart electric mobility platform articulated around three integrated components: a fleet of

connected electric scooters, a network of IoT-enabled physical docking stations, and a cross-platform mobile application.

1.2 General Overview of Transportation in Algeria

1.2.1 Organisation of the Transportation System

Algeria's transport system has evolved through three identifiable phases that are useful for understanding its present configuration. From independence in 1962 until 1988, the State exercised an essentially monopolistic role over both passenger and freight transport, with public enterprises absorbing the totality of the offer. A first liberalisation step was taken in 1988 with Law 88–17, which opened private investment to road passenger transport, in a context where the State was no longer in a position to fully finance the expansion of public networks. The current legal framework was established by Law 01–13 of 7 August 2001 on the orientation and organisation of land transport (LOOTT), which consolidated the principle of a mixed economy in the sector and clarified the respective competences of the central administration, the wilayas, and the municipalities [6, 5].

Within this framework, three layers of actors share the operational responsibility for urban transport: the Ministry of Transport, which formulates national policy, manages the regulatory framework, and supervises public enterprises; the wilaya-level *Direction des Transports de la Wilaya* (DTW), which acts as the deconcentrated operational organiser, attributing exploitation licences, coordinating modal interconnections, and supervising compliance; and finally the operators themselves — public enterprises (ETUSA for Algiers urban buses, EMA for the Algiers metro, SNTF for rail), private bus operators that emerged after the 1988 liberalisation, and state-licensed urban taxis [1, 3]. The institutional architecture is summarised in Figure 1.1.

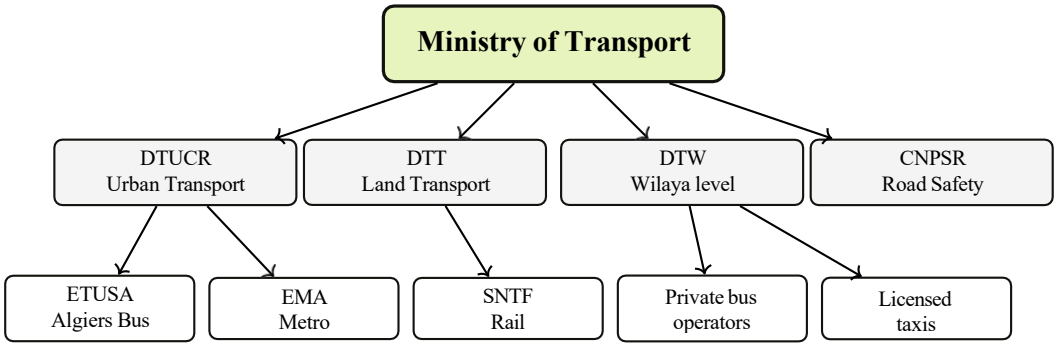


Figure 1.1. *Institutional architecture of the Algerian transport system.*

In parallel to this hierarchical structure, several other ministries intervene transversally: the Ministry of Public Works for road infrastructure, the Ministry of Housing and Urbanism for spatial planning, the Ministry of Interior and Local Authorities for municipal coordination,

and the Ministries of Commerce and Finance for tariff regulation and budget allocation [1, 7]. This multiplication of stakeholders is one of the recurring difficulties of urban transport governance, and has been identified as a structural factor in the coordination gaps observed in major metropolitan areas [5].

The transport offer itself can be visualised along a historical axis (Figure 1.2). The three legislative phases described above are punctuated by major infrastructural milestones: the inauguration of the Algiers metro in 2011, the deployment of seven tramway networks in major cities (Algiers, Oran, Constantine, Sidi Bel Abbès, Ouargla, Sétif, Mostaganem) between 2011 and 2023, and the ongoing modernisation of the national rail network operated by SNTF [3].

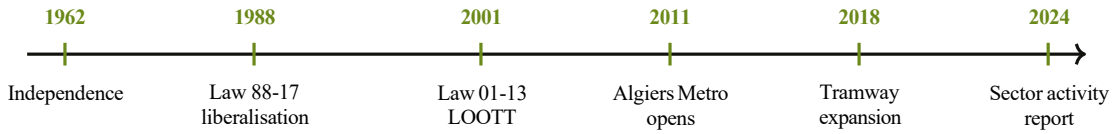


Figure 1.2. *Historical timeline of major reforms and infrastructural milestones of the Algerian transport sector.*

1.2.2 Importance of Urban Mobility

Urban mobility is not merely a technical question of moving people from one point to another; it is a strategic determinant of economic productivity, social cohesion, and environmental sustainability. Comparative analyses of Maghreb countries indicate that mobility inefficiencies cost between 2% and 5% of GDP annually through unproductive time, fuel overconsumption, and indirect productivity losses [8, 2]. For Algeria, where the transport sector already absorbs a substantial share of household and public expenditure, this represents a non-trivial macroeconomic opportunity.

At the social level, mobility conditions access to fundamental rights and opportunities. A citizen who cannot reach an employer, a university, or a healthcare facility within reasonable cost and time becomes structurally disadvantaged. This dimension is particularly visible for women, the elderly, and lower-income populations, whose access to private vehicles is statistically lower and who therefore rely disproportionately on public and shared modes [9, 1].

The official transport-sector activity report for the first half of 2024 provides a measure of the current scale of the system: the seven Algerian tramway networks transported a combined 73.7 million passengers, while the Algiers metro added another 23.3 million [3]. Substantial as they are, these figures mask important spatial and temporal heterogeneities: mass-transit infrastructure remains concentrated in a small number of metropolitan areas, leaving secondary cities and most university campuses dependent on less structured solutions. Table 1.1 summarises the modal split of public ridership at the national level in 2024.

Table 1.1. *Estimated passenger volumes of major Algerian public transport modes, first half 2024.*

Mode	Passengers (H1 2024, millions)	Geographic coverage
Tramway (7 cities)	73.7	7 metropolitan areas
Metro (Algiers only)	23.3	1 line, Algiers only
Air transport (internal)	3.6	36 airports
Air transport (international)	4.5	36 airports
Maritime (passengers)	0.242	7 maritime stations
Rail (SNTF)	est. 8–10	National network
Urban bus (ETUSA + private)	est. 350–400	All urban areas

Beyond the raw volumes, the strategic significance of urban mobility lies in its multiplier effects on all other economic sectors. A reliable transport network reduces the marginal cost of labour mobility, expands the geographic perimeter within which firms can recruit, supports tourism and commerce, and contributes to environmental policy objectives by reducing per-capita emissions when alternatives to the private car are available. The absence or weakness of such a network produces the opposite cascade: labour-market segmentation, depressed commercial productivity, and locked-in dependency on individual motorisation [8].

1.3 Transportation Problems in Algeria

This section examines, mode by mode, the structural problems that affect Algerian urban mobility. The objective is not to enumerate all known deficiencies, but to identify those that produce the operational void within which the *Rafeeq* system is designed to operate.

1.3.1 Bus Transportation

The collective bus system constitutes the backbone of public mobility in Algerian cities, absorbing approximately 80% of all motorised public-transport demand in the metropolitan area of Algiers, and a comparable share in secondary cities [1, 2]. Despite this central role, the service suffers from a cluster of structural weaknesses that have been consistently documented over the last two decades.

Operating schedules are rigid, with off-peak intervals routinely exceeding 20 to 30 minutes, which discourages spontaneous trips and forces users to organise their movements around bus passages rather than around their own needs. During peak hours, vehicles are systematically overloaded, regularly exceeding twice their nominal seated capacity, which compromises both comfort and safety, particularly for women, elderly persons, and students (Figure 1.3).

Figure 1.3. *Overcrowding on an Algerian public bus during peak hours. The systematic mismatch between fleet capacity and peak demand degrades comfort and discourages routine use.*

Coverage gaps further amplify the problem. Bus lines are dimensioned to serve trunk corridors between major urban poles, leaving large portions of residential neighbourhoods, university campuses, and peripheral districts inadequately served [5]. The aging of the public fleet adds a third vulnerability: the average vehicle age exceeds twelve years in many wilayas, generating frequent mechanical failures, high pollutant emissions, and a degraded perceived service quality. Finally, service availability decreases markedly on weekends, public holidays, and evening hours — precisely when alternative modes are also less accessible.

Quantitatively, the gap between offer and demand is non-trivial. Field studies of the Algiers urban network indicate that, during morning peak hours, an estimated 35% to 45% of intending users either cannot board the first arriving bus or choose not to board because of overcrowding [1, 2]. This unmet demand translates either into time loss (waiting for the next

bus) or into modal shift towards more expensive alternatives such as taxis and private cars. The cascading effect is well-known in transport economics under the name of the **vicious circle of public transport**: as buses lose riders to private cars, road congestion increases, buses become slower, and they lose still more riders [10].

1.3.2 Private Car Transportation

The private automobile has become the dominant motorised mode in Algerian cities, with more than 6.5 million vehicles registered nationally and an annual growth rate of approximately 8% over the last decade [9]. This growth has outpaced road infrastructure expansion, which has progressed by less than 2% annually over the same period, producing a struc-tural imbalance between vehicle volume and road capacity. The motorisation rate, while still lower than in European reference countries, has risen rapidly: between 2010 and 2023, the ratio of private vehicles per thousand inhabitants in Algeria increased from approximately 110 to over 145, a 32% rise that has been almost entirely absorbed within urban areas.

Figure 1.4. *Chronic traffic congestion on a main artery of an Algerian metropolitan area. Average peak-hour speeds on such corridors hover around 12 km/h.*

The consequences of this imbalance are visible daily. Average peak-hour speeds on Algiers' main arteries hover around 12 km/h, comparable to a brisk walking pace [2]. Finding a legal parking spot in central districts typically requires 15 to 25 minutes of search, which frequently leads drivers to park illegally and, in turn, further reduces effective road capacity. A 2022 observational study in central Algiers estimated that during midday hours, between 18% and 28% of all moving traffic on observed corridors consisted of vehicles actively searching for parking, a phenomenon that produces no productive trip-output yet contributes substantially to congestion and emissions [9].

The full ownership cost of a private car — including depreciation, fuel, insurance, taxes, maintenance, and parking — renders short-distance use disproportionately expensive. Table 1.2 decomposes the effective cost per kilometre of a mid-range urban-use sedan in the Algerian context.

Table 1.2. *Estimated cost breakdown of private car ownership for urban use (mid-range sedan, Algerian context).*

Cost component	**Annual cost (DZD)**	**Cost per km (DZD)**
Depreciation (10-year amortisation)	120,000	16
Fuel (8 L/100 km at 47 DZD/L)	28,200	4
Insurance (mandatory + complementary)	24,000	3
Annual technical inspection and taxes	6,000	1
Routine maintenance and repairs	40,000	5
Parking (when paid)	12,000	2
Total (assuming 7,500 km/year)	**230,200**	**31**
Marginal cost of a 2 km urban trip	—	*70–110 (incl. search/parking)*

Beyond the household level, the private car is the dominant source of urban CO_2 emissions. The transport sector accounts for approximately 25% of national emissions, and within transport, private cars dominate the urban share [8]. The carbon intensity of an individual short-distance trip is particularly high relative to its utility: transporting a single person of approximately 70 kg in a 1,500 kg vehicle imposes an energy and emission overhead of more than 95% on every kilometre travelled, which constitutes one of the strongest physical arguments in favour of lightweight individual mobility modes [11].

1.3.3 Taxi Transportation

Taxis — including state-licensed urban taxis and informal collective taxis (commonly referred to as *fraudaires*) — fill several niches that buses cannot cover, particularly on-demand and door-to-door trips. They are, however, subject to limitations that prevent them from serving as a reliable everyday mode for short urban trips. The fare structure makes short trips economically unattractive for the driver: a typical 2 km trip in Algiers costs the passenger between 150 and 250 DZD, while the driver's revenue per kilometre is lower than for longer journeys. This economic logic produces the widely observed phenomenon of drivers refusing short fares, particularly during peak hours [5, 7].

Figure 1.5. Citizens waiting for a taxi during peak hours. The combination of high demand and selective acceptance of short fares produces effective waiting times that frequently exceed those of public buses.

Even when a driver accepts a short fare, the user faces additional friction. Outside the metered urban taxis available only in a handful of cities, fare negotiation introduces social discomfort and price unpredictability. Waiting times in residential neighbourhoods or during off-peak hours can reach 20 to 40 minutes, which negates the advantage of on-demand availability. Asking a taxi to wait during a short shopping or administrative stop is operationally inefficient and culturally awkward, since the driver loses revenue while parked.

The economic geometry of taxi service for short trips is structural. The fixed cost of accepting a fare — vehicle depreciation associated with stopping, fuel consumed during deceleration and acceleration, and the opportunity cost of forgoing a longer fare — is largely independent of trip length, while the revenue scales with distance. The break-even distance below which a fare becomes unattractive lies around 3 to 4 km in the current Algerian tariff regime; the structural consequence is that the supply of taxis is naturally biased *against* the trip-length profile that is most needed by First and Last Mile users [12].

1.3.4 Walking Commute

Walking remains the most ecological, economical, and health-promoting urban mobility mode for very short distances. In the Algerian context, however, several structural and environmental factors restrict its usability for daily commutes beyond approximately 1 to 1.5 km. Pedestrian infrastructure is uneven across the territory: many neighbourhoods lack continuous sidewalks, or have sidewalks obstructed by parked vehicles, informal markets, or construction debris (Figure 1.6) [1]. The Mediterranean climate, while temperate on annual average, includes summer periods where daily temperatures regularly exceed 35°C and brief but intense winter rain episodes; both reduce the appeal of walking distances of 2 to 3 km.

Figure 1.6. *Pedestrian environment in an Algerian urban district, illustrating typical obstructions to walking as a daily transport mode.*

Safety considerations add a further layer of complexity. In peripheral neighbourhoods and university districts, inadequate street lighting and the absence of dedicated pedestrian crossings increase both the perceived and the actual risk of walking, particularly at night and especially for women. The cumulative effect is that walking remains realistic only for trips below approximately 1.5 km, a threshold that excludes the majority of student, work, and commercial trips in dispersed Algerian urban geographies [7].

A complementary perspective is provided by the temporal economy of walking: a 2 km walk at the average urban pace of 4.8 km/h requires about 25 minutes, which represents an opportunity cost that grows superlinearly with distance because the marginal disutility of each additional minute increases as the trip lengthens. Combined with the carrying constraints inherent to walking — the practical impossibility of transporting bags of groceries, laptops, or other items above a few kilograms over comparable distances — this disutility curve explains why walking is rapidly substituted by motorised alternatives even for relatively short trips [13].

1.4 Urban Traffic Congestion Problem

1.4.1 Causes of Congestion

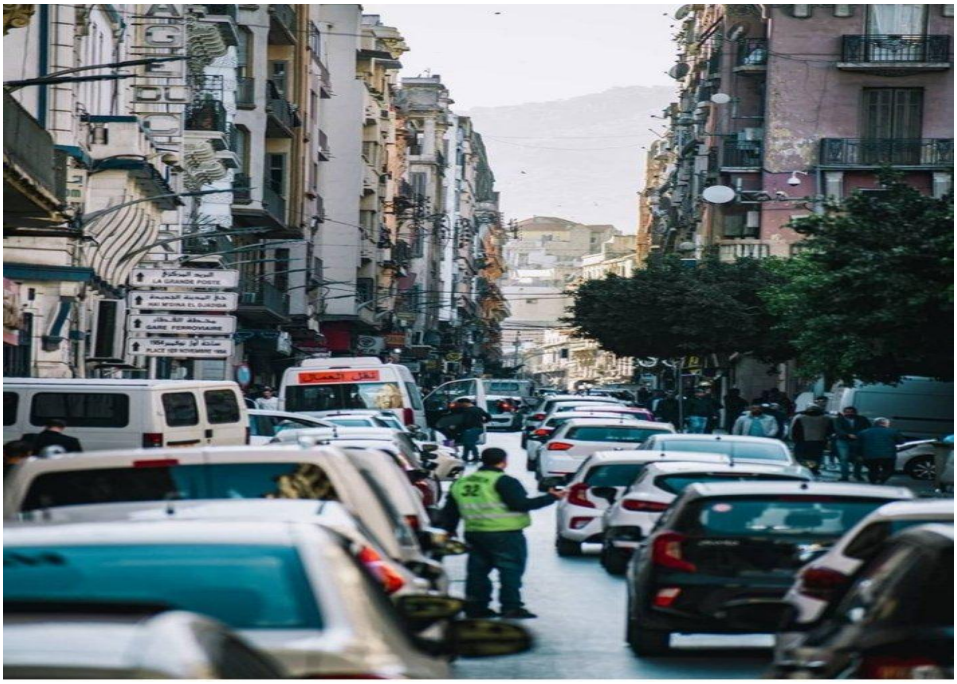


Figure 1.7. *Saturated parking conditions in a central Algerian urban district. Illegal on-street parking is estimated to reduce effective roadway capacity by 30% to 50% along central corridors.*

Urban congestion in Algeria is not a transient phenomenon but the cumulative outcome of structural decisions taken over several decades. Five principal causal factors interact, summarised in Figure 1.8. The first is rapid motorisation without commensurate infrastructure investment, with the private vehicle fleet growing roughly four times faster than the road network. The second is the centralised character of Algerian urban planning: economic, administrative, and educational functions are concentrated within compact city centres, generating radial commuting flows that converge on a limited number of arterial roads during predictable peak windows.

The third factor is the limited capacity of mass-transit alternatives outside the capital: only Algiers possesses a metro system, and tramway coverage, although expanding, serves a small fraction of the national urban population. The fourth is the absence of physical separation between traffic categories: pedestrians, bicycles, scooters, buses, and trucks share the same road space, reducing the operational efficiency of each mode. The fifth and often underestimated cause is illegal on-street parking, which by some estimates reduces effective roadway capacity by 30% to 50% along central corridors [5, 2].

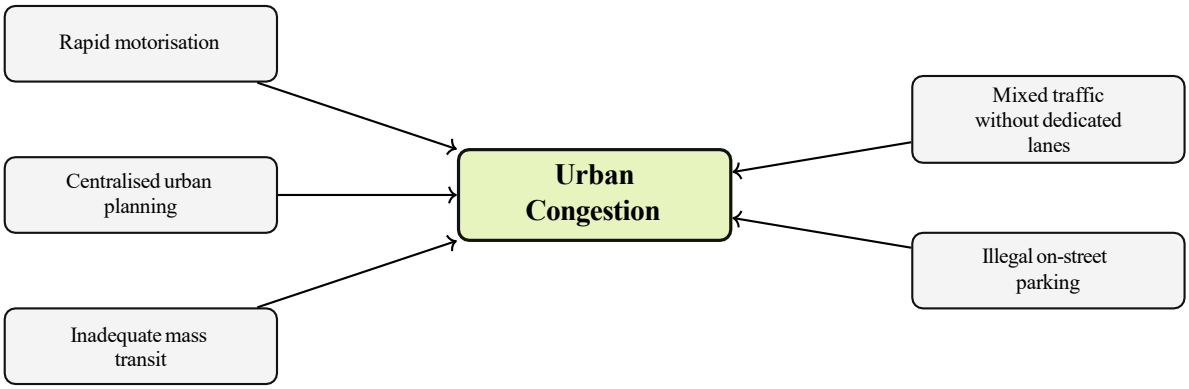


Figure 1.8. *Principal structural causes of urban congestion in Algerian cities.*

The legislative response has progressed in parallel. The draft of the new Algerian Road Traffic Code (2024) introduces stricter penalties for illegal parking and dangerous driving, formalises the role of municipalities in traffic-flow planning, and includes provisions related to the management of light electric vehicles — a category that explicitly covers electric scooters of the kind used by *Rafeeq* [14].

1.4.2 Consequences: Time, Pollution, Stress

The consequences of chronic congestion are multidimensional and impose costs at the individual, collective, and environmental levels. Table 1.3 synthesises these dimensions.

Table 1.3. *Synthesis of the consequences of urban congestion in Algeria.*

Dimension	Typical manifestation
Time loss	~240 hours/year per Algiers commuter (30 working days equivalent)
Air pollution	NO_x and $PM_{2.5}$ regularly above WHO thresholds in Algiers, Oran, Constantine
Public health	Respiratory disease prevalence elevated along main arteries
Psychological	Elevated stress, anxiety, aggressive driving behaviour
Economic	2% to 3% of urban GDP lost annually
Social	Regressive impact on those without private vehicles

Figure 1.9. *Cascading effects of urban congestion on air quality and public health. The transport sector accounts for approximately 25% of national CO_2 emissions.*

Time loss is the most immediately perceived consequence: the average Algiers commuter loses approximately 240 hours per year in congestion, equivalent to thirty full working days [2]. Air quality is compromised by elevated concentrations of nitrogen oxides and fine particulate matter, both of which regularly exceed WHO recommended thresholds in major Algerian metropolitan areas and are causally linked to respiratory and cardiovascular morbidity [8]. Beyond physical consequences, prolonged exposure to congestion produces measurable psychological impacts, including elevated stress, anxiety, and aggressive driving behaviour.

The aggregate economic cost is estimated at 2% to 3% of urban GDP annually, accounting for productivity losses, fuel overconsumption, and accelerated vehicle wear [2]. Finally, congestion produces a regressive social effect: citizens without private vehicles — typically students, young workers, and lower-income households — bear a disproportionate share of mobility difficulties, since they depend on public transport whose operational performance is itself degraded by congestion.

A useful synthesis is provided by the relation between trip distance and effective unit cost across modes (Figure 1.10). For trips below 3 km, the cost per kilometre of every existing mode increases sharply: buses become disproportionately slow because of overcrowding and frequency mismatch; taxis become economically infeasible because of the minimum-fare structure; private cars accumulate disproportionate fixed costs over short distances; walk-ing becomes impractical beyond physical and climatic thresholds. This region of the cost–distance curve constitutes the **operational void** within which the First and Last Mile problem

is situated.

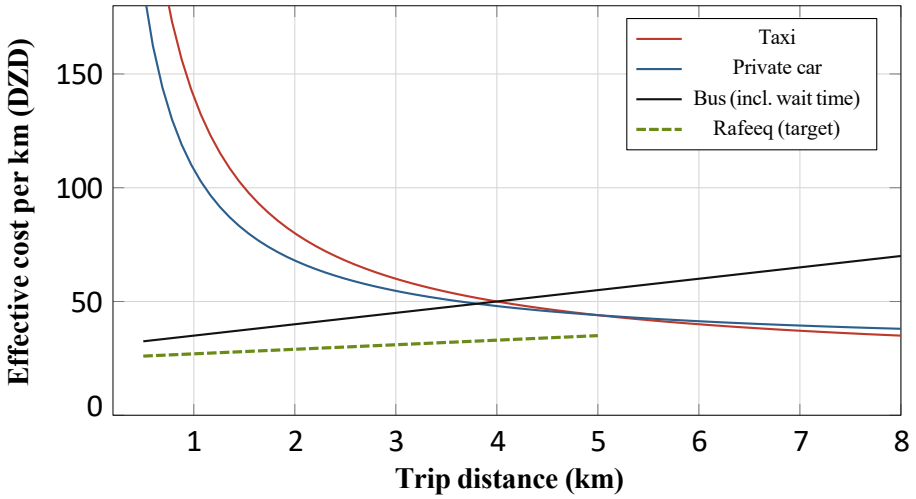


Figure 1.10. *Effective cost per kilometre as a function of trip distance, by mode. The 1–3 km region exhibits high marginal costs for all conventional modes, defining the operational void targeted by Rafeeq.*

1.5 The First and Last Mile Problem

1.5.1 Definition

The **First and Last Mile** (FLM) problem is a well-established concept in transportation planning that refers to the difficulty of covering the initial and final portions of a multi-modal trip [12, 13]. Originally formulated in the context of freight logistics, the term has been progressively extended to passenger transport to describe the gap between a citizen's origin or destination and the nearest mass-transit node. These segments, although geographically short (typically 0.5–3 km), often represent the most inconvenient portion of a journey because they fall below the threshold at which mass-transit modes are efficient yet exceed the threshold of comfortable walking.

In a passenger context, the FLM problem manifests in three principal configurations: travel from the user's home to the nearest stop of a structured transport network; travel from the arrival stop to the final destination (workplace, school, shopping centre); and inter-modal transfer between two transport modes, such as bus to taxi or train to bus. In all three cases, the FLM segment is the structural bottleneck of the broader trip: a long-distance bus or rail journey may be efficient on its trunk segment, but the overall user experience is determined by the worst of the chained segments, which is typically the FLM portion [15, 16].

1.5.2 Impact on Students and Citizens

The FLM problem has disproportionate impact on populations with limited transport options. Students must traverse distances between university residences and academic departments — often 1 to 3 km — multiple times per day, typically under tight time constraints imposed by class schedules. Workers face FLM challenges when connecting from public transport stops to industrial zones or office complexes located on urban peripheries. Ordinary citizens encounter FLM frictions during shopping, medical appointments, social visits, and administrative procedures. The cumulative effect is a daily friction that reduces overall mobility, limits opportunities, and degrades quality of life.

The impact is not equally distributed across demographic groups. Women, who are statistically less likely to hold a driving licence or to own a private vehicle in the Algerian context, depend more heavily on public transport and walking, and therefore absorb a disproportionate share of FLM frictions in their daily routines [1, 7]. Elderly citizens, for whom walking distances above one kilometre are physiologically demanding, are effectively excluded from large portions of the urban territory when no intermediate-range mode is available. Lower-income households face a compounded constraint: the modes that would resolve their FLM problem (taxis, private cars) are precisely those whose unit cost they cannot sustain on a routine basis, while the affordable modes (buses, walking) are precisely those that fail to cover the FLM segment. The structural consequence is a regressive accessibility gradient that reinforces existing inequalities in access to education, employment, and healthcare [9, 8].

1.5.3 Case of University Campuses in Algeria

Algerian universities provide an especially clear illustration of the FLM problem. The national higher-education system comprises 116 universities and equivalent institutions serving more than 1.8 million students. The majority of these students live in publicly managed university residences (*cités universitaires*) located between 1 and 3 km from their academic departments, with the National Office of University Works (ONOU) operating a free shuttle-bus service connecting the two [4].

In practice, the ONOU service exhibits the same structural limitations as urban public transport: rigid schedules with frequencies of 30 to 45 minutes during peak periods, overcrowding that exceeds nominal capacity, and limited operating hours that exclude evening and weekend academic activities. Walking remains feasible for shorter distances but becomes impractical when students must transport books, laptops, or shopping, particularly in adverse weather. Taxis are too expensive for repeated daily use within a student budget, and personal vehicles are accessible only to a small minority. Figure 1.11 visualises the typical mobility pattern of an Algerian student and the FLM gap that emerges within it.

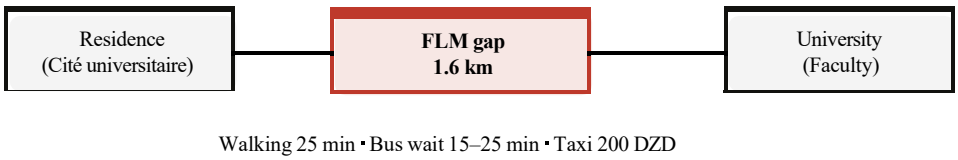


Figure 1.11. *The First and Last Mile gap on a typical Algerian university campus.*

The University of Mohamed El Bachir El Ibrahimi in Bordj Bou Arreridj, which constitutes the institutional context of the present work, hosts approximately 18,000 students. The main university residence is located approximately 1.6 km from the central campus and is served by a fleet of about twelve ONOU shuttle-buses. Peak-hour waiting times of 15 to 25 minutes are routinely followed by 5 to 10 minutes of travel in overcrowded condi-tions, translating into a daily time loss of 30 to 50 minutes per student. Multiplied across the academic year, this represents roughly 120 hours of cumulative loss per student, which by any reasonable accounting constitutes a significant impediment to academic productivity and well-being [4, 7].

1.5.4 Empirical Validation: Survey Results

To empirically validate the operational void identified in the preceding sections, the authors conducted an online survey between March and May 2026 through a Google Forms instrument distributed via social media, university mailing lists, and direct outreach in the Bordj Bou Arreridj area. The survey included twenty-two questions covering demographics, current mobility habits, perceived friction points, willingness to adopt a smart-mobility solution, payment preferences, and operational concerns. At the closure of data collection, the consolidated dataset comprises **105 valid responses**, sufficient to identify statistically meaningful trends in the target population.

The respondent demographics confirm that the survey captured the intended target population. University students aged 18–24 dominate the sample at 68.6 %, with an additional 21.0 % in the 25–34 age band that broadly corresponds to early-career graduates; older respondents (35–54) account for 5.8 % of the sample, and respondents under 18 represent 4.8 %. Women account for 40.0 % of respondents and men for 60.0 %, a balance broadly consistent with the gender composition of the Algerian student population. By occupation, 60.0 % are university students, 21.0 % are employees, 11.4 % are currently unemployed, and 6.7 % are self-employed.

The mobility profile of the sample establishes the structural conditions that the First and Last Mile problem affects. Seventy point five per cent (70.5 %) of respondents do not own a private vehicle and rely on public transport for their daily mobility, with an additional 7.6 % owning only a bicycle or scooter; only 9.5 % own and use a car daily. The dominant short-distance mobility mode is walking (35.2 %), followed by the bus (30.5 %), the taxi (17.1 %), and the private car (12.4 %). More than 53 % of respondents perform three to five short trips

daily, and 62.8 % spend less than 100 DZD per day on transport, which establishes both the volume and the price sensitivity of the addressable demand.

Key Validation Results — Survey (n=105)

- **The First and Last Mile problem is widely recognised.** Forty-four point eight per cent (44.8 %) of respondents face the residence–campus FLM gap on a daily basis, an additional 14.3 % face it occasionally, and 16.2 % know someone who does. Only 19.0 % report no exposure to the problem.
- **Traffic congestion and bus waiting time dominate the friction.** Fifty-six per cent (56.2 %) cite traffic congestion as a major mobility problem, 54.3 % cite excessive bus waiting time, 48.6 % cite bus overcrowding, and 36.2 % cite taxi cost.
- **Taxi refusal of short trips is structural.** Forty per cent of respondents who use taxis (26.7 % several times, 13.3 % once or twice) have been refused a ride explicitly because the distance was deemed too short.
- **Daily time lost is substantial.** 52.4 % of respondents lose between 10 and 30 minutes per day to mobility problems, an additional 25.7 % lose between 30 and 60 minutes, and 8.6 % lose more than an hour.
- **Adoption intent for *Rafeeq* is strong.** 92.4 % are open to trying the service (61.0 % definitely, 31.4 % probably); 45.7 % would use it more than twice a day, and an additional 19.0 % at least once daily.
- **Price sensitivity is moderate.** 31.4 % expect a price below 3 DZD per minute, 21.0 % accept 3–5 DZD, 17.1 % accept 5–7 DZD, and 23.8 % report that comfort outweighs price entirely; these results confirm that the planned tariff structure of approximately 5 DZD per minute is within the acceptable envelope for the majority of the addressable demand.
- **Hybrid payment is critical.** CIB/Dahabia cards (67.6 %) and BaridiMob (59.0 %) dominate the preferred payment methods, with phone-shop top-up (29.5 %) and cash recharge kiosks (28.6 %) attracting substantial residual demand from users without bank access. No single channel dominates exclusively, confirming the necessity of a hybrid payment infrastructure.
- **Theft is the principal adoption barrier.** 47.6 % of respondents identify fear of theft or damage responsibility as their primary concern, 37.1 % cite road safety, and 19.0 % report not knowing how to ride a scooter; this finding directly motivates the layered anti-theft architecture documented in Chapter 3.

- **The momentary-stop feature is widely valued.** 88.5 % of respondents react positively to the idea of pausing a ride for a short stop without ending the rental (69.5 % will use it often, 19.0 % sometimes), confirming the operational value of the feature.
- **Time saving is the dominant motivator.** 76.2 % of respondents cite time saving as the principal reason that would lead them to try *Rafeeq*, 60.0 % cite avoidance of traffic congestion, 54.3 % cite money saving, 40.0 % cite the curiosity of trying a new mobility format, and 21.0 % cite contribution to environmental protection.
- **Highest-value station locations.** Universities and residence halls (79.0 %), major bus terminals (58.1 %), shopping centres (55.2 %), residential neighbourhoods (54.3 %), and hospitals plus government administrations (49.5 %) constitute the priority deployment locations identified by the respondents.

The survey thus provides direct empirical support for the central thesis of this chapter: an operational void exists, citizens perceive it daily, and a credible majority is ready to adopt a properly designed alternative. The dominance of the student demographic in the sample, combined with the very high adoption intent and the explicit identification of universities as the highest-value deployment site, also concretely supports the choice of the University of Mohamed El Bachir El Ibrahimi campus as the location of the planned pilot deployment. The complete survey instrument, demographic breakdown, and per-question response distributions are presented in Appendix A.

1.6 Limitations of Existing Solutions

1.6.1 Inadequacy of Public Transport

The principal limitation of public bus transport in resolving the FLM problem is not operational but structural. Bus systems are dimensioned for mass movement along fixed routes at scheduled intervals; they are therefore fundamentally incompatible with the FLM use case, which requires individualised, on-demand, short-distance service [5, 10]. No incremental improvement of frequencies or route density can fully reconcile these two logics: increasing frequency raises operating costs disproportionately to the marginal demand on FLM segments, while densifying routes dilutes service quality on trunk corridors. The university shuttle-bus systems operated by ONOU exhibit the same intrinsic limitation: even with significant additions of vehicles, they cannot match the on-demand profile that students require during their daily class schedule.

1.6.2 High Cost of Taxis

Taxis provide the on-demand flexibility that buses lack, but their cost structure makes them unsuitable for routine multi-trip daily use. A student undertaking two short daily trips at an average cost of 150 DZD each would accumulate 300 DZD per day, or approximately 6,000 DZD per month over twenty active days. For an average student grant of approximately 2,000 DZD per month, this represents three times the entirety of the disposable income, which is economically and socially impossible [9, 4]. For working adults, the comparable share of monthly income is lower in absolute terms but still represents 15% to 20% of an average salary, which similarly precludes routine use.

1.6.3 Constraints of Personal Vehicles

Personal vehicles, when available, are poorly adapted to short urban trips for both economic and operational reasons. The fixed costs of ownership (vehicle acquisition, insurance, taxes, depreciation) are incurred regardless of usage intensity, so the per-trip cost of short trips remains high. The variable costs (fuel and parking) further amplify this effect: using a 1,500 kg vehicle to transport a single person over 2 km is energetically and environmentally inefficient and is moreover penalised by parking search times that often exceed actual driving time. From a systemic perspective, the marginal cost of one additional private vehicle on a congested corridor is much higher than its individual cost, since each vehicle contributes to the congestion experienced by all other users [8, 11].

1.6.4 Synthesis: The Operational Void

Combining the analyses above, the operational profile required to address the FLM problem can be characterised by four simultaneous conditions: (1) availability on demand, with a maximum waiting time of a few minutes; (2) unit cost per trip below approximately 100 DZD; (3) accessibility for citizens without bank accounts or smartphones with international payment instruments; and (4) door-to-door operation with minimal parking friction. Figure 1.12 situates each existing mode in the two-dimensional space defined by the first two conditions and demonstrates that none of the current alternatives populates the region required by the FLM use case.

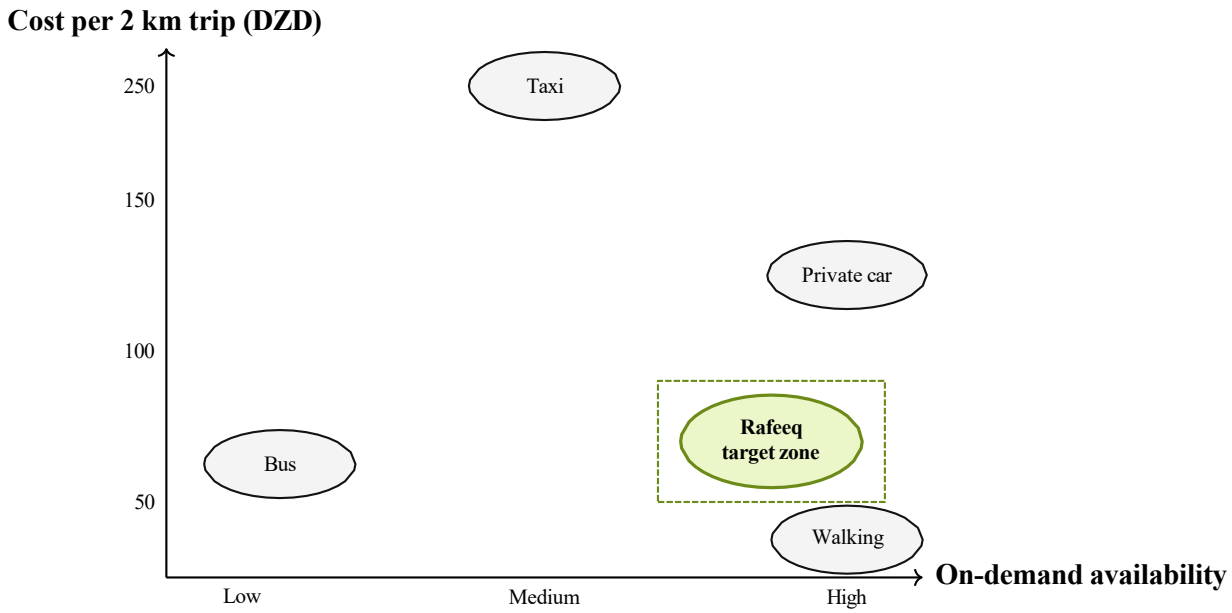


Figure 1.12. The operational void in the on-demand-availability × cost plane. Existing modes leave the region of low cost and high on-demand availability unoccupied, which is precisely the region targeted by Rafeeq.

Key insight

This diagrammatic synthesis reinforces the central thesis of the chapter: the FLM problem is not addressable by incremental improvement of any existing mode, but requires the introduction of a new mode whose operational profile is fundamentally different.

1.7 Proposed Solution: Rafeeq

1.7.1 Overview of the Rafeeq System

Rafeeq (Arabic for "companion") is a smart electric mobility platform designed to address the FLM problem within the specific institutional, economic, and cultural context of Algerian cities. The system integrates three operational components: a fleet of connected electric scooters, a network of physical docking stations equipped with IoT-controlled electromechanical locks, and a cross-platform mobile application that orchestrates the user interaction. Figure 1.13 summarises these three components and their interactions.

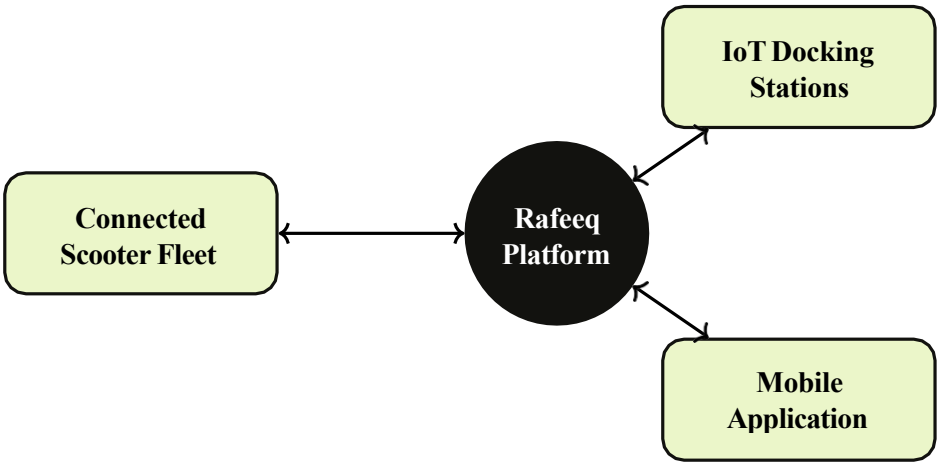


Figure 1.13. *The three integrated components of the Rafeeq platform.*

The intended user experience can be described as follows. A citizen approaches a docking station, identifies an available scooter through the mobile application, scans its QR code, and is granted physical access within seconds. The trip is then performed door-to-door, with the application displaying live cost and speed. Upon arrival, the user docks the scooter at any *Rafeeq* station, where the IoT-controlled lock physically secures the vehicle. The total cost is computed automatically and debited from the user's prepaid wallet. The entire interaction is intermediated digitally; no cash, paper ticket, or human operator is required at the point of use.

1.7.2 Project Objectives

The *Rafeeq* project pursues four interdependent objectives.

(1) Resolve the First and Last Mile problem by offering affordable, on-demand, short-distance transport with a service profile that no existing mode currently fills in the Algerian context.

(2) Contribute to the reduction of urban congestion by encouraging a modal shift from private vehicles and taxis to light electric vehicles for short trips.

(3) Promote financial inclusion through a hybrid payment system that combines bank cards (CIB, Dahabia), the mobile payment service BaridiMob, cash recharge kiosks at major stations, and a network of partner retail shops, thereby making the service accessible to citizens without bank accounts.

(4) Demonstrate the operational feasibility of advanced IoT infrastructure in the Algerian context, establishing a technical and organisational foundation that can be extended to other smart-city domains.

1.7.3 General Architecture (Mobile, Backend, IoT)

The *Rafeeq* system is architected as three vertically integrated layers (Figure 1.14). The mobile layer is implemented as a cross-platform Flutter application supporting Arabic (with right-to-left orientation), French, and English. The cloud backend is a FastAPI service hosted on Render and persists state in a PostgreSQL database; it exposes thirty-two REST endpoints and maintains WebSocket channels for real-time communication with both mobile clients and IoT endpoints. The IoT layer is bifurcated to reflect the distinct operational profiles of the two endpoint types: each scooter embeds a Raspberry Pi 3 Model B+ paired with a SIM800L 2G GPRS modem that together communicate with the cloud backend and with the scooter's electronic speed controller (ESC) over Bluetooth Low Energy (BLE) using the Nordic UART service, while each docking station embeds an ESP32 microcontroller that drives an electromechanical Solenoid lock via a TIP122 power stage.

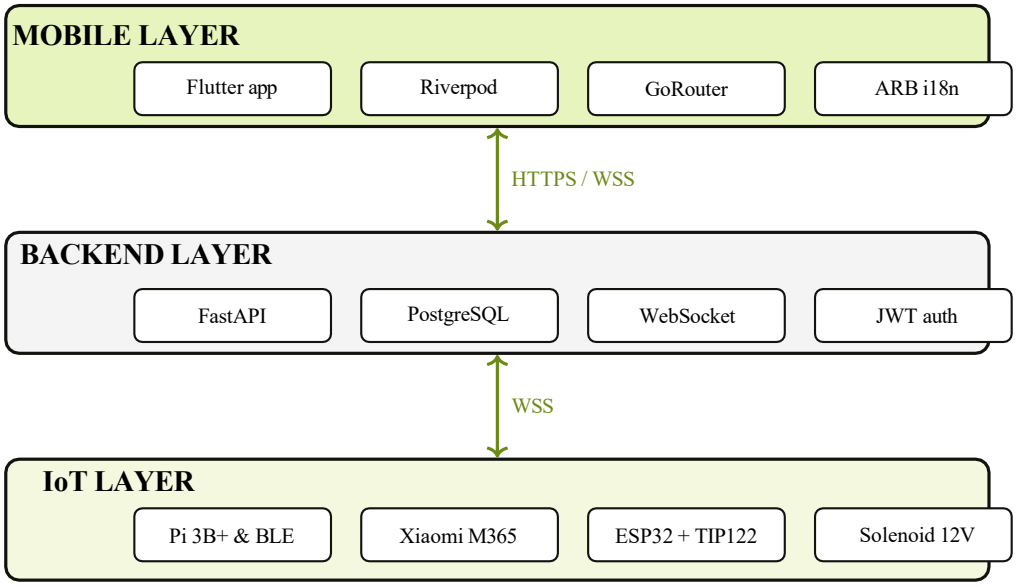


Figure 1.14. *Layered architecture of the Rafeeq platform.*

This vertical decomposition supports independent evolution of each layer and isolates technological risk: a change in the scooter model affects only the IoT layer; a change in payment provider affects only the backend; a change in user-interface guidelines affects only the mobile layer. The detailed engineering of these layers is the subject of Chapters 2 and 3.

1.7.4 Docking Stations (Security and Organisation)

The choice to anchor *Rafeeq* on physical docking stations, rather than adopting a free-floating model, is one of the defining design decisions of the project. International experience with free-floating scooter and bicycle systems has documented three recurring difficulties: high rates of vandalism and theft, which can reach 20% of fleet per year in some European cities; visual and functional disorder on sidewalks, which generates complaints from residents and

pedestrians; and significant operational costs associated with the daily collection, charging, and redeployment of dispersed vehicles [16, 17].

A docked system addresses each of these issues structurally. Physical anchoring through an electromechanical lock reduces vandalism by removing the opportunistic component of theft. Concentration of vehicles at designated stations preserves urban order and integrates the system visually into the public space. Charging contacts embedded in the docks eliminate the need for manual collection of vehicles by operations teams. The docking design therefore aligns the technical, urban, and economic dimensions of the service, at the cost of an initial capital investment in stations that is amortised over the operational life of the fleet.

1.7.5 Advantages Over Existing Solutions

A comparative assessment of *Rafeeq* relative to the existing transport modes on the typical 1–3 km use case is summarised in Table 1.4. The criteria correspond to the dimensions identified as critical in the preceding sections: on-demand availability, cost per trip, waiting time, door-to-door access, parking requirement, environmental footprint, financial inclusion, and resilience to congestion.

Table 1.4. *Comparative analysis of urban transport modes for trips of 1–3 km.*

Criterion	Bus	Taxi	Private car	Rafeeq
On-demand availability	Low	Medium	High	High
Cost per 2 km trip (DZD)	30–50	150–250	80–120	40–60
Average wait time	15–30 min	5–15 min	0 min	1–3 min
Door-to-door access	No	Yes	Yes	Yes
No parking needed	Yes	Yes	No	Yes
CO_2 emissions	Medium	Medium	High	Very low
Financial inclusion	Cash only	Cash only	N/A	Card + Cash
Suitable in congestion	Poor	Poor	Very poor	Excellent

Rafeeq occupies a structurally distinct position in this comparison. It combines the on-demand flexibility of the taxi, the cost profile of public transport, the environmental footprint of walking, and the institutional integration that distinguishes it from informal alternatives. The hybrid payment system addresses, specifically and explicitly, the financial inclusion gap that is rarely accounted for in international shared-mobility comparisons but is decisive in the Algerian context, where a substantial share of citizens does not hold a bank card or use mobile banking applications [8, 3].

1.8 Conclusion

This chapter has established the contextual and analytical foundation for the *Rafeeq* project. The Algerian transport system, despite considerable public investment since the early 2000s,

remains structurally inadequate to address short-distance on-demand urban mobility. None of the existing modes — public buses, taxis, private vehicles, or walking — can simultaneously satisfy the four criteria that define the FLM use case: availability on demand, affordability for daily multi-trip use, accessibility for citizens without bank accounts, and environmental efficiency.

The FLM problem manifests across virtually all urban Algerian contexts but is particularly acute in university campuses, where students bear a recurring time and financial cost imposed by the structural mismatch between fixed-route shuttle systems and individual study schedules. The University of Mohamed El Bachir El Ibrahimi in BBA, used as illustrative case study throughout this work, exhibits the typical configuration: a residence–campus distance of approximately 1.6 km, a shuttle-bus service whose operational profile leaves substantial gaps in availability, and a student population whose budget excludes routine taxi use.

The *Rafeeq* system has been introduced as an integrated response to this configuration. It articulates three components — a connected scooter fleet, IoT-enabled docking stations, and a cross-platform mobile application — along a vertically decomposed architecture that supports incremental deployment, independent evolution of layers, and explicit adaptation to the Algerian institutional, economic, and cultural context. The comparative analysis demonstrates that *Rafeeq* occupies a position in the mode–criterion space that is currently unoccupied by any existing solution.

The remaining chapters of this thesis translate this conceptual proposition into a concrete engineering deliverable. The next chapter formalises the functional requirements and the global system architecture using the APTE methodology and justifies the technological choices that guided implementation. The final chapter details the design, fabrication, software implementation, and experimental validation of the three subsystems — including the electronic design of the docking station, the BLE communication protocol with the scooter, the mobile application screens, and the end-to-end latency measurements obtained on the real prototype.

CHAPTER 2

Functional Analysis and Architecture of the Rafeeq System

2.1 Introduction

Chapter 1 established the contextual and analytical foundation upon which the *Rafeeq* project is built, and identified the operational void that the system is designed to fill. The present chapter translates that conceptual proposition into a structured engineering deliverable. The objective is twofold: to formalise the functional requirements that the system must satisfy, and to derive a global architecture that supports those requirements while accommodating the technological, economic, and operational constraints of the Algerian context.

The methodology adopted in this chapter combines two complementary engineering traditions. The first is the **APTE methodology** (*Application aux Techniques d'Entreprise*), a French functional-analysis framework standardised in NF X50-151 that decomposes a system through three canonical representations: the need diagram (*bête à cornes*), the interactor diagram (*diagramme pieuvre*), and the FAST tree [18, 19, 20]. The second is the **Unified Modeling Language** (UML), an internationally standardised object-oriented modelling notation that complements APTE with use-case, sequence, state, and class diagrams [21]. The combined methodology provides both the structural rigour required by engineering supervision and the behavioural granularity required by software implementation.

The remainder of the chapter is organised as follows. Section 2.2 formalises the requirements specification (*cahier des charges*) of *Rafeeq* along six dimensions: main functions, constraint functions, technical, ergonomic, energy, and economic-security constraints. Section 2.3 applies the APTE methodology and produces the three canonical functional diagrams. Section 2.4 complements the analysis with four UML representations covering use cases, sequence flows, state transitions, and the underlying data model. Section 2.5 presents the global system architecture, decomposed into five subsystems and the communication flows that interconnect them. Section 2.6 justifies the technological choices that guided the implementation, comparing each selected option against viable alternatives. Section 2.7 synthesises the chapter and outlines the engineering deliverables developed in Chapter 3.

2.2 Requirements Specification

The requirements specification of *Rafeeq* is structured along six dimensions that collectively define the operational envelope within which the system must perform. The decomposition follows the format recommended by NF X50-151 for functional analysis, distinguishing between **main functions** (FP, *fonctions principales*) that express the value delivered to the user, and **constraint functions** (FC, *fonctions contraintes*) that capture the conditions imposed by the operational environment [20].

2.2.1 Main Functions (FP)

Three main functions express the fundamental service that *Rafeeq* provides to its users. They are formulated in the active form, with an identified actor, a directly affected object, and a measurable purpose.

FP1 — Provide on-demand short-distance mobility. The system enables a user to obtain physical access to an electric scooter within a maximum of three minutes from the moment the request is initiated through the mobile application, and to travel between any two docking stations of the network at an average commercial speed of 15 to 25 km/h.

FP2 — Transparently bill the trip and the wallet. The system computes the cost of every trip on a per-minute tariff structure, deducts the corresponding amount from a prepaid user wallet, and issues a digital receipt at the end of each trip. All financial operations are auditable and reconciliable with the operator's accounting records.

FP3 — Secure the scooter when not in use. The system physically immobilises every scooter outside an active rental session by means of an IoT-controlled electromechanical lock anchored in the docking station, thereby preventing unauthorised removal, theft, and vandalism.

2.2.2 Constraint Functions (FC)

Eight constraint functions formalise the conditions that the system must satisfy in order to operate reliably and acceptably within its target environment.

FC1 — Adapt to the Algerian regulatory framework, including Law 01–13 on land transport, the draft Road Traffic Code provisions for light electric vehicles, and the technical regulations issued by the Ministry of Transport [6, 14].

FC2 — Operate under Mediterranean climatic conditions, including ambient temperatures from -5°C to $+45$°C, occasional rainfall, and sustained sunlight exposure.

FC3 — Support hybrid payment instruments, including bank cards (CIB, Dahabia), the mobile-money service BaridiMob, and cash recharge for citizens without bank accounts.

FC4 — Communicate over heterogeneous network conditions, including 2G GPRS cellular coverage of intermittent quality for the mobile scooter uplink (provided by SIM800L modems) and Wi-Fi backhaul at fixed station sites.

FC5 — Be ergonomic for non-technical users, including students, elderly citizens, and first-time smartphone users, with a localised user interface in Arabic, French, and English.

FC6 — Be resistant to vandalism and theft, with mechanical and electronic protections appropriate to the public-space deployment context.

FC7 — Be energy-efficient, with scooter charging entirely supplied by the docking sta-tion and station electrical consumption compatible with the Algerian residential grid (220 V, 50 Hz).

FC8 — Be economically deployable at scale, with a per-station capital cost below the equivalent of three months of expected operational revenue at target ridership.

2.2.3 Technical Constraints

Beyond the functional formulation above, several technical constraints derive directly from the choice of physical and software components. The scooter platform must be compatible with the Xiaomi M365 BLE protocol, whose specification has been documented through community reverse engineering [22]. The docking station must drive an electromechanical Solenoid lock with a holding force of at least 50 N and a release latency below 500 ms; the lock must operate in a Fail-Secure mode (i.e. remain locked in the absence of power) to prevent silent theft during outages. The scooter must implement a layered anti-theft policy combining the original Xiaomi front-wheel electronic brake, the station-side captive ring, and a GPS-anchored displacement watchdog that triggers a local audible alarm and a backend alert when displaced more than fifteen metres from the rest position. The cloud backend must support concurrent WebSocket connections from every active scooter and station, with a target end-to-end latency below 300 ms for unlock commands. The mobile application must operate on Android 8.0 and above, the dominant version base among Algerian smartphone users.

2.2.4 Ergonomic and User Experience Constraints

The user-facing dimension of the system imposes ergonomic constraints whose satisfaction conditions adoption. The mobile application must support full right-to-left layout in Arabic, with localised currency formatting in Algerian Dinars and Hijri-Gregorian date conversion. The on-boarding flow must be completable in less than 90 seconds for a first-time user, including the KYC step required by Algerian financial regulation. The unlock interaction must be triggerable from a single gesture (QR scan or proximity tap), and the trip-end interaction

must be triggerable with at most two confirmations. The interface must be readable in direct sunlight and operable with gloves, two conditions that influence the typography and the minimum touch-target size.

2.2.5 Energy Constraints

The energy budget of the system is shaped by the dual constraint of operational autonomy and grid compatibility. Each scooter is fitted with a 36 V, 7.8 Ah lithium-ion battery delivering a nominal range of approximately 25 km per charge under standard conditions; this corresponds to roughly fifteen average urban trips between recharges. The docking station provides a charging current of up to 2 A per scooter through a contact interface integrated in the dock. The station itself draws between 60 and 120 W during continuous charging cycles, which falls within the operating envelope of an ordinary residential circuit and avoids dedicated grid reinforcement. The Raspberry Pi 3 Model B+ onboard the scooter is powered by the scooter battery through a dedicated DC-DC converter and consumes approximately 2.5 W in normal operating conditions and 0.4 W in idle, with an additional 1 W contribution from the SIM800L modem during continuous transmission.

2.2.6 Economic and Security Constraints

Two final dimensions condition the deployability of the system in the Algerian context. Economically, the capital cost of each docking station must remain below $180 USD per scooter slot in order to reach payback within twenty-four months at the target ridership of four daily rentals per slot. Securely, the system must satisfy the data-protection principles set out in Algerian Law 18–07 on the protection of personal data, including encryption of personally identifying information at rest and in transit, segregation of payment data from operational data, and configurable retention periods. All communications between subsystems use Transport Layer Security (TLS 1.3), and JWT-based authentication is enforced on every API endpoint [23].

Table 2.1 synthesises the requirements specification along the six dimensions described in this section and provides the valuation criteria that will be used in Chapter 3 to assess the prototype.

Table 2.1. *Synthesis of the Rafeeq requirements specification.*

Dimension	Requirement	Target
Functional	Unlock latency (request to release)	≤ 3 s
Functional	Door-to-door commercial speed	15–25 km/h
Functional	Wallet billing precision	per-minute
Technical	End-to-end command latency	≤ 300 ms
Technical	Solenoid release time	≤ 500 ms
Technical	Android API level	≥ 26 (8.0)
Ergonomic	Onboarding completion time	≤ 90 s
Ergonomic	Supported languages	ar, fr, en
Ergonomic	Minimum touch target	48×48 dp
Energy	Scooter autonomy per charge	≥ 20 km
Energy	Station continuous power draw	≤ 120 W
Energy	Pi 3B+ idle/operating consumption	0.4 / 2.5 W
Economic	Station capital cost per slot	≤ $180 USD
Economic	Target payback period	≤ 24 months
Security	Transport encryption	TLS 1.3
Security	Authentication	JWT (RS256)
Security	Anti-theft displacement threshold	15 m
Security	Device authentication (IoT)	32-byte token

2.3 APTE Functional Analysis

The APTE methodology proceeds in three structural stages. The need diagram expresses the raison d'être of the system in a single sentence. The interactor diagram (*pieuvre*) identifies the external environments with which the system interacts and the functions that arise from these interactions. The FAST diagram decomposes the principal functions into solutions through a recursive "How? / Why?" interrogation, providing the bridge between functional requirements and technical architecture.

2.3.1 Need Diagram (Bête à Cornes)

The need diagram answers three questions about the system: to whom does it render service, on what does it act, and to what purpose [18, 20]. For *Rafeeq*, the formalisation is given in Figure 2.1. The system renders service to the urban Algerian citizen — specifically the segment requiring on-demand short-distance transport that no existing mode satisfies, as established in Chapter 1. The object on which it acts is the citizen's mobility itself: the system transforms a structurally unaddressed mobility need into an effective, billable, and physically secured trip. The purpose is to resolve the First and Last Mile problem within the institutional, economic, and cultural context of Algerian cities.

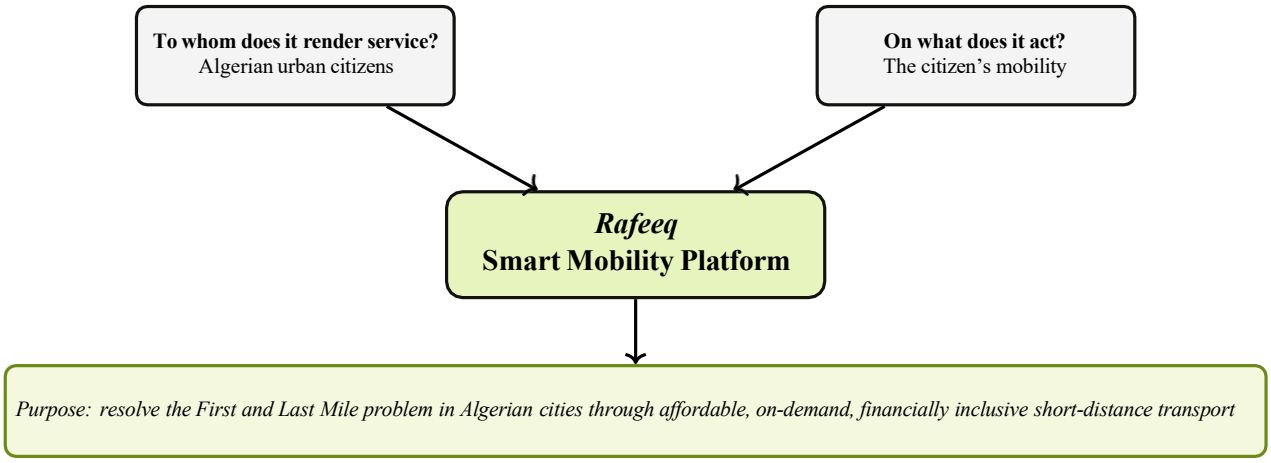


Figure 2.1. Need diagram (bête à cornes) of the Rafeeq system.

2.3.2 Octopus Diagram (Diagramme Pieuvre)

The interactor diagram identifies the external environments (*milieux extérieurs*) with which the system interacts and derives the principal and constraint functions from these interactions [18, 19]. Each main function (FP) connects two external environments through the system, while each constraint function (FC) connects the system to a single environment. Figure 2.2 presents the *Rafeeq* pieuvre with eight identified milieux.

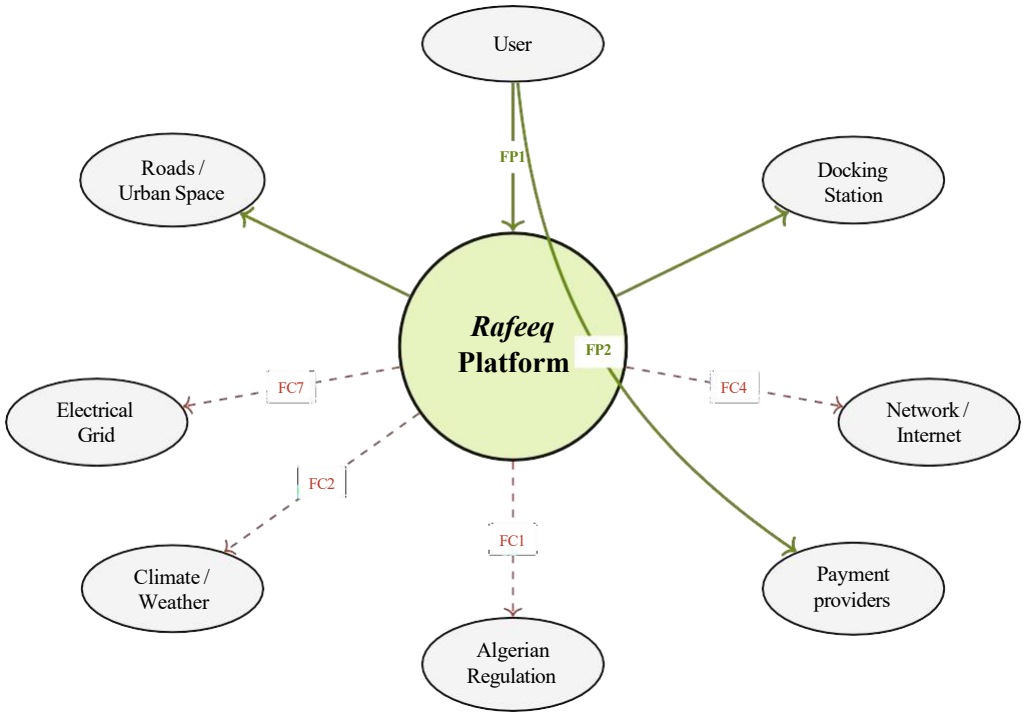


Figure 2.2. Octopus diagram (pieuvre) of the Rafeeq system, with the principal main and constraint functions identified by their FP/FC code.

2.3.3 Functional Specifications Table

The functions extracted from the pieuvre diagram are formalised in the functional specifications table (Table 2.2), which associates each function with a quantitative criterion, an acceptance level, and a degree of flexibility (F0 = imperative, F1 = important, F2 = negotiable). This table constitutes the contractual interface between the analytical formulation of the requirements and their technical implementation; every test in Chapter 3 can be traced back to an entry of this table [20].

Table 2.2. *Functional specifications: main functions (FP) and constraint functions (FC) of the Rafeeq system.*

Code	Function	Criterion	Level	Flex.
FP1	On-demand short-distance mobility	Unlock latency	≤ 3 s	F0
FP1	Commercial speed	Average urban speed	15–25 km/h	F1
FP2	Transparent billing	Tariff resolution	per-minute	F0
FP2	Wallet update latency	End-to-end	≤ 5 s	F1
FP3	Scooter immobilisation	Solenoid holding force	≥ 50 N	F0
FP3	Anti-theft displacement threshold	GPS-anchored watchdog	15 m	F1
FC1	Regulatory compliance	Algerian laws referenced	01-13, CR-2024	F0
FC2	Climatic operation	Temperature range	−5 to +45 °C	F0
FC3	Hybrid payment	Number of payment modes	≥ 3	F0
FC4	Network adaptability	2G GPRS minimum bandwidth	≥ 40 kbps	F1
FC5	User ergonomics	Onboarding time	≤ 90 s	F1
FC6	Anti-vandalism	IP rating	≥ IP54	F1
FC7	Energy efficiency	Station continuous power	≤ 120 W	F1
FC8	Economic deployability	Cost per slot	≤ $180 USD	F2

2.3.4 FAST Diagram

The FAST (Function Analysis System Technique) diagram decomposes each main function into elementary functions, then maps each elementary function to a technical solution through the recursive interrogation “How is this function realised?”. Reading right to left provides the inverse interrogation “Why is this technical solution present?”, which serves as a validation pass against the original requirements. Figure 2.3 presents the FAST decomposition of FP1 (*Provide on-demand short-distance mobility*), which is the central function of the system.

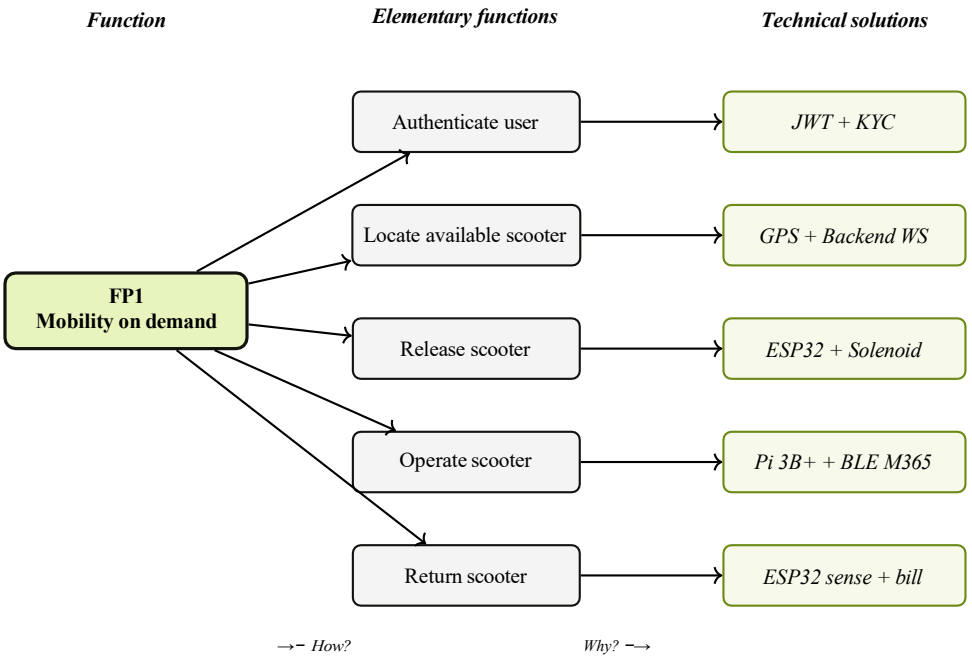


Figure 2.3. FAST decomposition of FP1 (Provide on-demand mobility) of the Rafeeq system.

Key insight

The APTE analysis translates the abstract problem identified in Chapter 1 into a concrete set of measurable functions and constraints. Every technical decision in the following sections is traceable back to a row of Table 2.2 or a branch of Figure 2.3.

2.4 System Modeling (UML)

Where the APTE methodology captures the structural and functional dimensions of the system, UML provides the behavioural and dynamic dimensions necessary for software de-sign [21]. This section presents four complementary UML representations: a use-case di-agram identifying actors and their interactions with the system, a sequence diagram detailing the scooter unlock flow, a state diagram modelling the scooter lifecycle, and a class diagram defining the underlying data model.

2.4.1 Use Case Diagram

The use-case diagram identifies the actors that interact with *Rafeeq* and the use cases through which these interactions are realised. Three principal actors are distinguished: the *Citizen*, who is the primary user of the rental service; the *Administrator*, who manages the fleet, the tariff structure, and the user accounts; and the *Maintenance Technician*, who handles station and scooter servicing. Three secondary actors complete the model: the *Payment Provider* (CIB, BaridiMob), the *Docking Station* (treated as an actor because it issues asynchronous

events such as lock-engaged or charge-complete), and the *Scooter* itself. Figure 2.4 formalises this model.

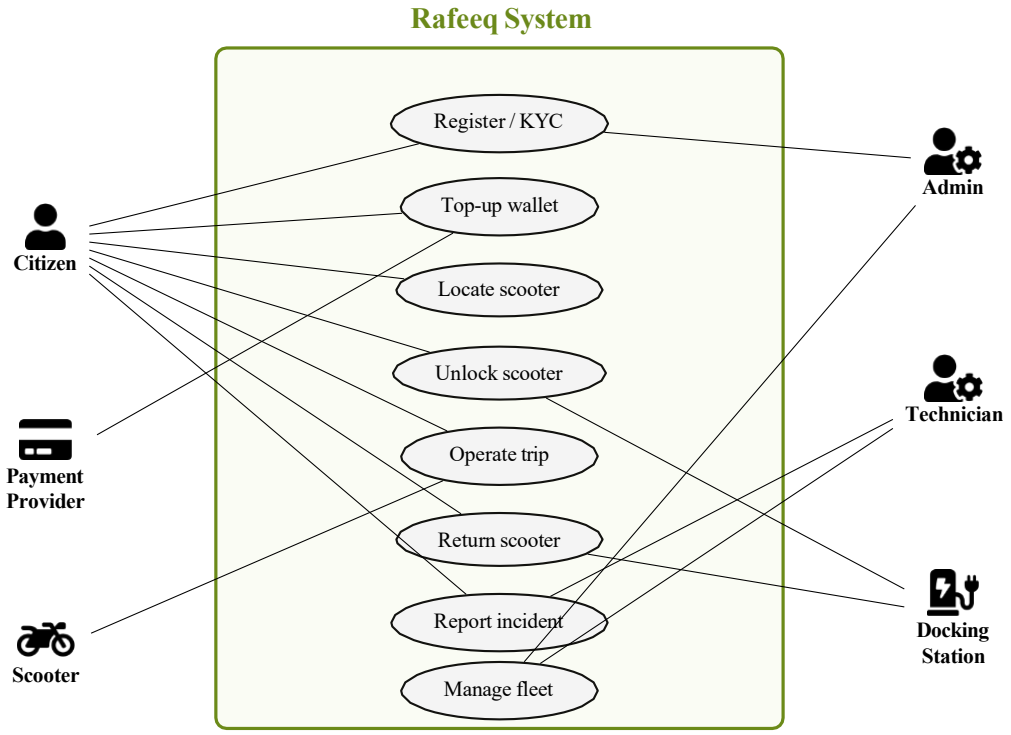


Figure 2.4. *UML use-case diagram of the Rafeeq system. Each ellipse represents an elementary use case; lines indicate the participation of actors in each use case.*

The use-case "Unlock scooter" (UC4) is the most operationally complex of the model because it chains four subsystems: the mobile application, the cloud backend, the docking station controller, and the scooter onboard computer. The next subsection details the corresponding sequence.

2.4.2 Sequence Diagram — Scooter Unlock Flow

Figure 2.5 models the message exchanges between the four participating subsystems when a citizen unlocks a scooter. The flow begins with a QR scan triggered by the citizen, propagates through the backend for authorisation and billing pre-check, and terminates with the BLE command issued by the Raspberry Pi to the Xiaomi M365 electronic speed controller. The target end-to-end latency for the entire sequence is 3 seconds.

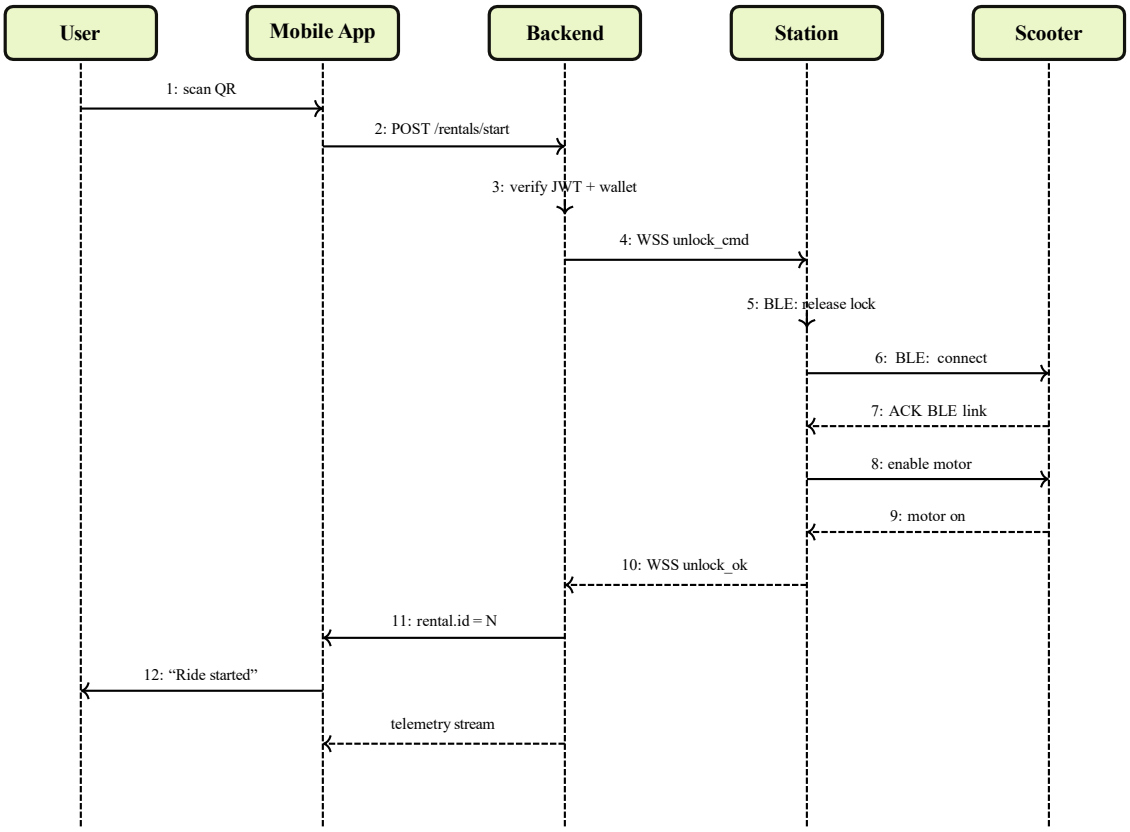


Figure 2.5. *UML sequence diagram of the scooter unlock flow. The target end-to-end latency from message 1 (QR scan) to message 12 ("Ride started") is 3 seconds.*

2.4.3 State Diagram — Scooter Lifecycle

A scooter in the *Rafeeq* fleet traverses a finite number of operational states throughout its lifecycle. Figure 2.6 models these states and the transitions between them as a UML state machine. The states are mutually exclusive at any instant and are persisted in the back-end database; transitions are triggered either by user actions (*rent*, *return*, *pause*) or by autonomous events (*battery_low*, *fault_detected*, *geofence_breach*, *theft_suspected*). The *theft_suspected* event is generated by the GPS-anchored anti-theft watchdog described in Chapter 3: when the scooter is in the *Available* or *Maintenance* state and its position drifts more than fifteen metres from the recorded anchor, the watchdog triggers a local audible alarm and a backend alert, and the scooter is flagged in a dedicated `theft_alerts` table for operator follow-up.

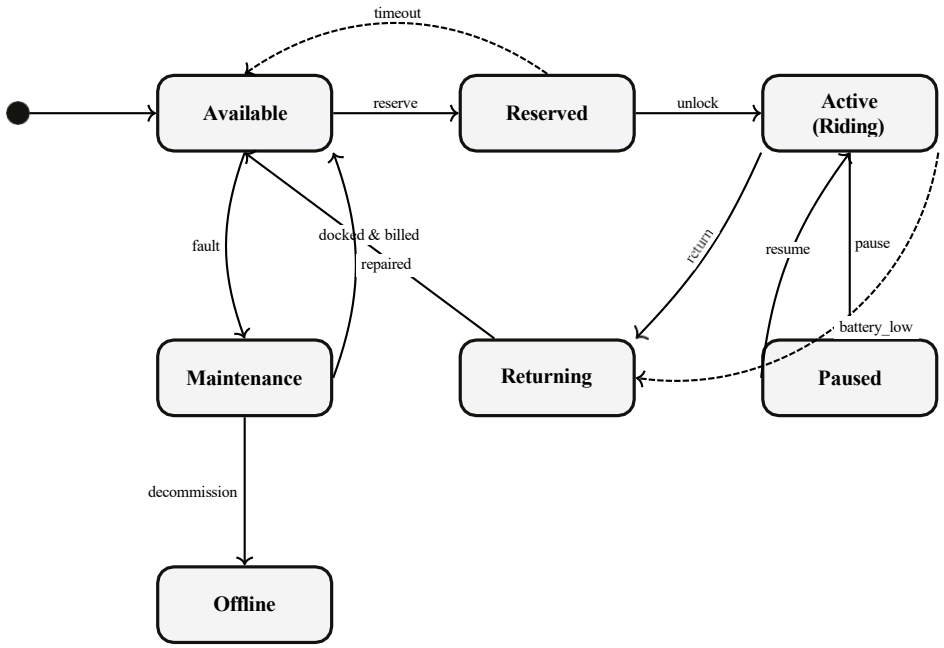


Figure 2.6. UML state machine modelling the scooter lifecycle in the Rafeeq fleet. Dashed arrows indicate autonomous transitions; solid arrows indicate user- or operator-driven transitions.

2.4.4 Class Diagram — Data Model

The persistent data model of the system is captured by the UML class diagram of Figure 2.7. Six core entities are identified: *User*, *Wallet*, *Trip*, *Scooter*, *Station*, and *Payment*. The associations between these entities follow the operational logic of the service: a user owns exactly one wallet, performs zero or more trips, and may register one or more payment instruments; a trip references one scooter and starts and ends at one station each; a station hosts a fixed number of scooter slots.

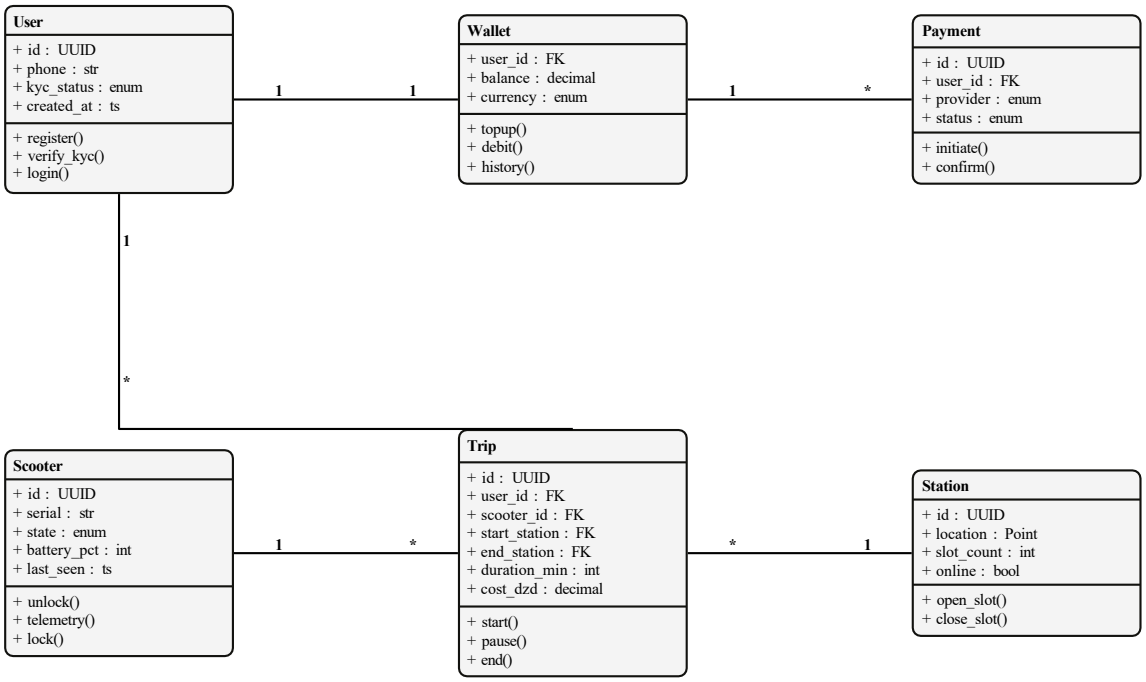


Figure 2.7. *UML class diagram of the Rafeeq data model. Multiplicities follow standard UML notation (1 = exactly one, * = many).*

2.5 Global System Architecture

The architectural decomposition of *Rafeeq* aligns with the three-layer model introduced in Section 1.7 (mobile, backend, IoT) and refines it into five interoperable subsystems. Each subsystem is independently developable, testable, and deployable, and exposes a well-defined interface to its neighbours. This section examines each subsystem in turn and concludes with a unified view of the communication flows between them.

2.5.1 Three-Layer Architecture (Extended)

Figure 2.8 presents the extended layered architecture. Compared with the simplified view of Chapter 1, this representation explicitly distinguishes the two IoT endpoint families (scooter and station), separates the backend into its functional modules (REST API, WebSocket gateway, database, authentication), and identifies the principal external services with which the system integrates.

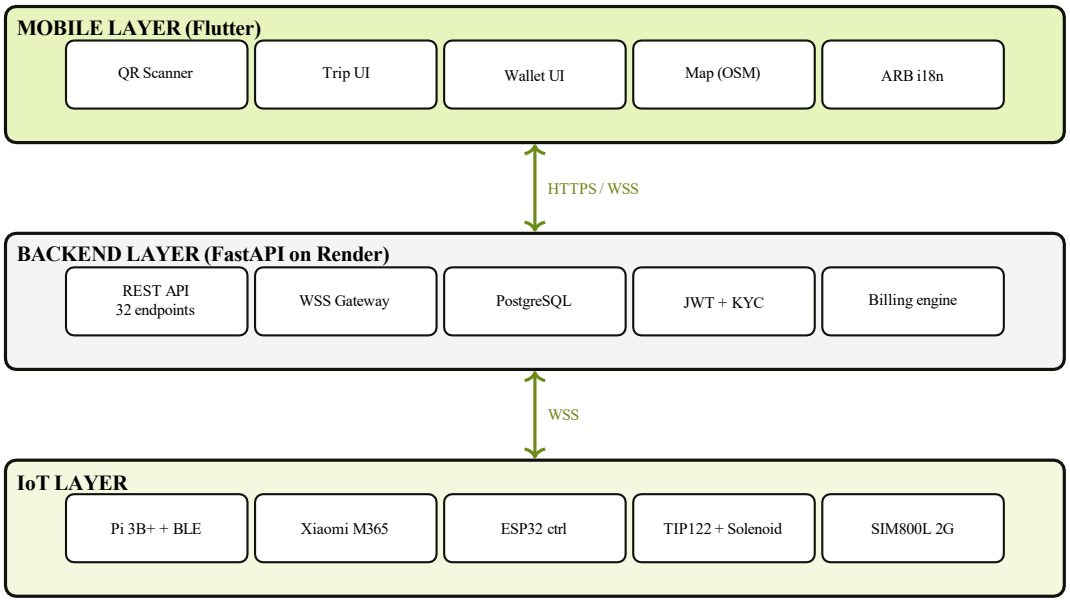


Figure 2.8. Extended three-layer architecture of the Rafeeq platform with the principal modules of each layer.

2.5.2 Scooter Subsystem

Each scooter integrates a Raspberry Pi 3 Model B+ single-board computer powered by the scooter battery through a dedicated buck converter. The Pi runs a Python daemon that maintains a persistent WebSocket Secure connection to the backend over a SIM800L 2G GPRS modem, and a Bluetooth Low Energy link to the scooter's electronic speed controller through the Nordic UART service. The Pi is responsible for unlock and lock commands, telemetry streaming (battery, speed, GPS position), and the enforcement of geofence limits. Figure 2.9 summarises the scooter electronics block.

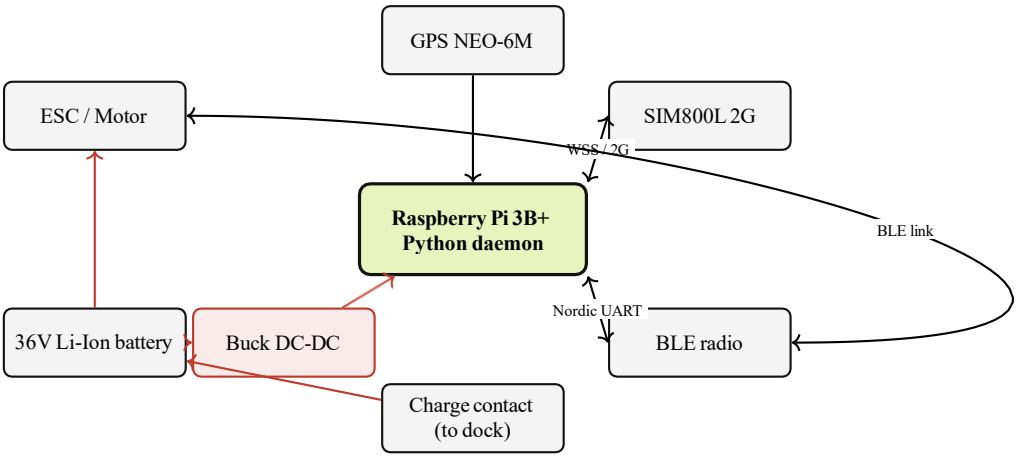


Figure 2.9. Block diagram of the scooter electronics subsystem. Red arrows indicate power paths; black arrows indicate data paths.

2.5.3 Docking Station Subsystem

The docking station hosts an ESP32 microcontroller that drives one electromechanical Solenoid lock per scooter slot through a TIP122 Darlington transistor used as a low-side switch [24]. The mechanical capture is implemented through a captive steel ring permanently attached to the scooter frame: when the scooter is docked, the ring slides into a slot on the side of the station, and the Solenoid pin — held in the extended position by an internal compression spring — passes through the ring and immobilises it. The de-energised state of the Solenoid is therefore the locked state (*Fail-Secure*), so that any failure of the controller, the network, or the power supply leaves the scooter secured rather than free. The ESP32 is responsible for the local lock/release logic, the maintenance of a persistent WebSocket connection to the backend over Wi-Fi, and the reporting of state transitions to the cloud. A 12 V Solenoid was selected for its low cost, mature supply chain, and compatibility with a single regulated rail. Figure 2.10 presents the station electronics block.

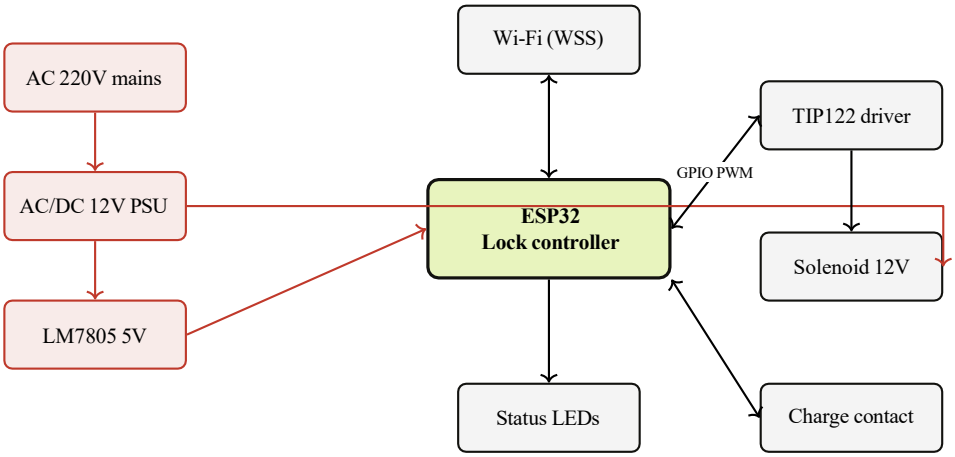


Figure 2.10. *Block diagram of the docking-station electronics subsystem. Red arrows indicate power paths; black arrows indicate data paths.*

2.5.4 Backend Subsystem

The cloud backend is a FastAPI application deployed on Render that orchestrates the entire platform. It exposes thirty-two REST endpoints organised into the functional groups summarised in Table 3.6. The backend persists state in a PostgreSQL 16 database and maintains two WebSocket gateways: one for IoT endpoints (lock commands, telemetry, theft alerts), and one for real-time trip events pushed to the mobile clients. Authentication is implemented through JSON Web Tokens with RS256 signatures and a refresh-token rotation policy [23, 25].

Table 2.3. REST endpoint groups exposed by the Rafeeq backend.

Group	Purpose	Endpoints
auth	registration, OTP, JWT issuance and refresh	6
kyc	document upload, status, verification	3
wallet	balance, top-up, transaction history	4
rentals	start, pause, resume, end, list	5
stations	list, detail, geo-search, slot occupancy	4
scooters	list, detail, telemetry, fault report	4
admin	fleet management, tariff, user moderation	6
Total	**REST surface**	**32**

2.5.5 Mobile Application Subsystem

The mobile application is a Flutter project organised around the Riverpod state-management framework and the GoRouter declarative router. The application supports right-to-left layout in Arabic, left-to-right layout in French and English, and uses ARB files for localisation [26, 27]. Sixteen screens cover the full user journey from on-boarding to trip history. Secure storage of the JWT token relies on the platform-specific keystore (Android Keystore on Android) through the flutter_secure_storage plugin; no security-relevant artefact is ever persisted to SharedPreferences. Figure 2.11 presents the internal module organisation.

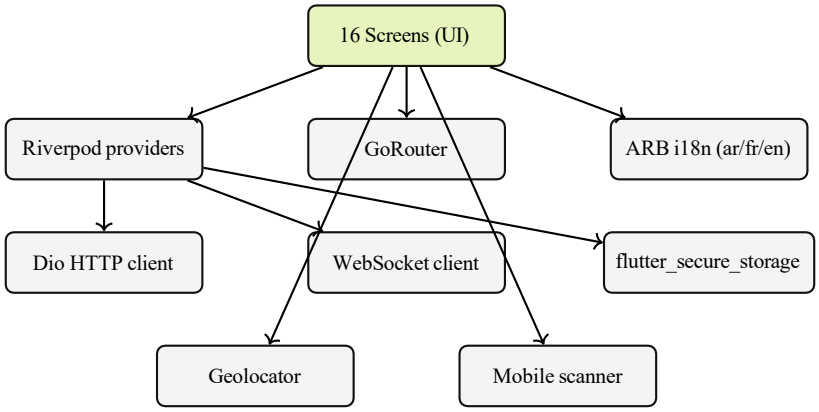


Figure 2.11. Internal architecture of the Flutter mobile application.

2.5.6 End-to-End Communication Flows

Figure 2.12 presents the complete communication topology of the platform. The mobile application communicates with the backend over HTTPS for REST calls and WSS for live trip telemetry. The scooter Raspberry Pi maintains a WSS link to the backend over a SIM800L 2G GPRS modem and a BLE link to the scooter ESC. The docking station ESP32 maintains a WSS link to the backend over Wi-Fi. The four subsystems are therefore federated through the backend, which acts as the single source of truth for the system state.

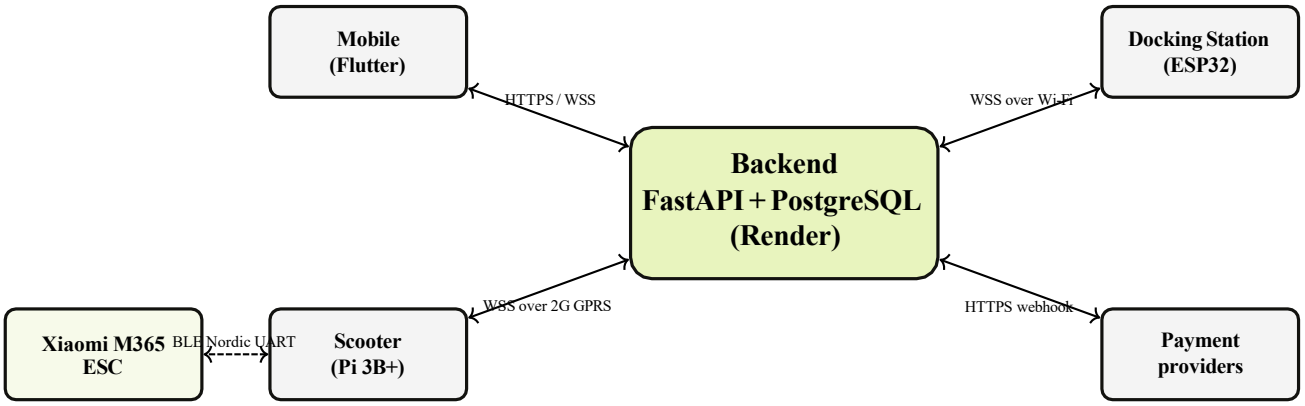


Figure 2.12. End-to-end communication topology of the Rafeeq platform.

2.6 Technological Choices

Every architectural decision documented above has been the outcome of a comparative evaluation. This section presents the trade-off analysis that motivated the selection of each principal technology, expressed as a comparison table against the most credible alternatives. The criteria of comparison are stable across tables: cost, ecosystem maturity, performance envelope, energy footprint, ease of integration in the Algerian context, and licence terms.

2.6.1 Scooter Onboard Computer — Raspberry Pi 3 Model B+

The scooter onboard computer must run a Linux distribution capable of hosting a Python daemon, must expose a usable BLE stack, and must remain operable in the thermal envelope of an outdoor enclosure. Table 2.4 compares the Raspberry Pi 3 Model B+ with two credible alternatives.

Table 2.4. Comparison of candidate scooter onboard computers.

Criterion	Pi 3B+ (1 GB)	ESP32-S3	STM32F7
CPU performance	4×1.4 GHz A53	2×240 MHz Xtensa	1×216 MHz M7
Linux + Python support	Native	ESP-IDF only	FreeRTOS / bare metal
BLE stack quality	Mature (BlueZ)	Good (NimBLE)	External chip required
Built-in BLE 4.2	Yes (onboard)	Yes	No
Cellular module integration	UART SIM800L	UART AT modem	UART AT modem
Power consumption (operating)	~2.5 W	~0.3 W	~0.2 W
Approx. unit cost (DZD)	6 000	1 500	5 000
Local market availability	Mature	Good	Limited
Selected	**Yes**	—	—

The Raspberry Pi 3 Model B+ was selected despite a higher unit cost and a heavier energy footprint than the microcontroller alternatives because of three decisive advantages. First, the Linux+Python combination dramatically reduces development time for a complex daemon

that must orchestrate BLE, GPS, telemetry, and WSS all in a single process, where the ESP32 and STM32 alternatives would have required a full re-engineering of the application on a bare-metal RTOS. Second, the Raspberry Pi ecosystem provides a mature toolchain for over-the-air updates and remote diagnostics, both critical for a fleet that will be deployed in the field [28]. Third, the scooter battery (36 V, 7.8 Ah) easily absorbs the 2.5 W steady-state consumption without materially affecting autonomy. The Pi 3B+ was preferred over the newer Pi 5 because of its lower unit cost, its lower power budget, and its immediate availability in the Algerian market.

2.6.2 Scooter Communication Protocol — BLE

The scooter onboard computer must control the Xiaomi M365 ESC, which exposes its command interface exclusively over Bluetooth Low Energy through the Nordic UART service. The choice of BLE is therefore largely imposed; nevertheless, Table 2.5 compares it against the alternatives that would have been viable if the protocol had been freely selectable.

Table 2.5. *Comparison of candidate scooter communication protocols.*

Criterion	BLE 5	Wi-Fi	LoRa
Range	10–50 m	30–100 m	1–15 km
Throughput	1–2 Mbps	50–150 Mbps	0.3–50 kbps
Power (TX)	5–15 mA	100–300 mA	30–90 mA
Pairing simplicity	Native	Captive portal	Complex (LoRaWAN)
Compatibility with Xiaomi M365	**Required**	—	—
Cost of radio chip	1 200 DZD	2 000 DZD	3 500 DZD
Selected	**Yes**	—	—

BLE provides the appropriate envelope for scooter-Pi communication: short range, low power, and a mature pairing model. The Nordic UART service, although nominally proprietary, has been documented through community reverse engineering and provides a stable interface to the M365 ESC [29, 30, 22].

2.6.3 Station Microcontroller — ESP32

The docking-station controller has a very different requirements profile than the scooter computer: it does not need a Linux stack, but it must drive the Solenoid GPIOs through the TIP122 power stage and maintain a permanent Wi-Fi WebSocket link with the cloud backend. Table 2.6 compares three candidates.

Table 2.6. Comparison of candidate station microcontrollers.

Criterion	ESP32-WROOM	Pi Zero 2 W	STM32F411
Wi-Fi built-in	Yes	Yes	External chip
Power consumption	80 mA	200 mA	30 mA + 80 mA Wi-Fi
GPIO availability	30+	26	50+
WebSocket stack maturity	ESP-IDF / Arduino	Linux native	FreeRTOS port
Approx. unit cost (DZD)	1 500	5 000	3 000
Selected	**Yes**	—	—

The ESP32 was selected for its native Wi-Fi, sufficient GPIO budget, mature Arduino/ESP-IDF ecosystem, and very low unit cost [31]. Its single-tasking model is sufficient for the station controller, whose responsibilities are essentially limited to GPIO actuation and WebSocket message exchange.

2.6.4 Backend Framework — FastAPI

The backend was the only subsystem whose technological choice was entirely free of physical constraints. Three candidate Python and Node-based frameworks were considered (Table 2.7).

Table 2.7. Comparison of candidate backend frameworks.

Criterion	FastAPI	Django	Express.js
Async support (native)	Yes	Partial (3.x)	Yes
Type-safe schema (Pydantic / TypeScript)	Yes	No	Optional
WebSocket support	Native (Starlette)	Channels (add-on)	Native
ORM integration	SQLAlchemy 2	Django ORM	Sequelize / Prisma
OpenAPI auto-generation	Yes	DRF add-on	Manual
Learning curve	Moderate	Steep	Easy
Selected	**Yes**	—	—

FastAPI was selected for three reasons. First, its native asyncio model elegantly supports the dual REST and WebSocket workload of *Rafeeq* without the architectural cost of a Django Channels overlay. Second, its Pydantic-based type validation produces an automatically generated OpenAPI schema, which is itself a deliverable item of the supervised project [25, 32]. Third, the framework's deployment footprint on Render is small, allowing the prototype to operate within a free or low-cost tier appropriate for a student-led startup.

2.6.5 Database — PostgreSQL

The relational data model of Figure 2.7 suggests a SQL database. Table 2.8 compares the most credible options.

Table 2.8. Comparison of candidate databases.

Criterion	PostgreSQL 16	MySQL 8	MongoDB 7
Relational integrity	Strong (ACID)	Strong (ACID)	Limited
Geospatial extensions	PostGIS	Spatial extension	Geo-indexes
JSON column support	Native (JSONB)	Yes (5.7+)	Native
Hosted on Render	Yes (managed)	External	External
Open-source licence	PostgreSQL Lic.	GPL	SSPL (restrictive)
Selected	**Yes**	—	—

PostgreSQL was selected because of its strict ACID compliance (critical for financial operations on user wallets), its mature PostGIS extension (useful for station and trip geo-queries), and the availability of a fully managed instance on Render [33].

2.6.6 Mobile Framework — Flutter

The mobile application targets both Android and iOS while sharing a single codebase. Table 2.9 compares Flutter with two reasonable alternatives.

Table 2.9. Comparison of candidate mobile frameworks.

Criterion	Flutter	React Native	Native (Kotlin/Swift)
Code sharing iOS/Android	95%+	80–90%	0%
Render performance	Native (Skia/Impeller)	Bridge overhead	Native
Right-to-left support	First-class	Good	First-class
Hot reload	Yes	Yes	No
APK size	7–12 MB	8–15 MB	4–8 MB
Team size required	1–2	1–2	2–4
Selected	**Yes**	—	—

Flutter was selected for three reasons. First, its 95%+ code sharing across platforms is a critical productivity factor for a three-person student team. Second, its first-class right-to-left layout support reduces the implementation cost of the Arabic localisation, which is the primary language of the target user base. Third, its rendering pipeline based on the Skia/Impeller graphics engine produces smooth animations on the mid-range Android devices that dominate the Algerian market [26].

2.6.7 Synthesis of Technological Choices

Table 2.10 summarises the technological stack and the rationale of every choice. Each row maps a subsystem to a chosen technology and to the principal alternative that was rejected.

Table 2.10. *Synthesis of the technological choices of the Rafeeq system.*

Subsystem	Chosen	Rejected	Decisive criterion
Scooter computer	Raspberry Pi 3B+	ESP32-S3	Linux+Python productivity
Scooter protocol	BLE 5	Wi-Fi	M365 ESC compatibility (imposed)
Station controller	ESP32-WROOM	Pi Zero 2 W	Native Wi-Fi + low cost
Backend framework	FastAPI	Django	Async + WebSocket + Pydantic
Database	PostgreSQL 16	MongoDB	ACID for wallet operations
Mobile framework	Flutter	React Native	Code sharing + RTL
Authentication	JWT (RS256)	Session cookies	Stateless backend, mobile-friendly
Transport security	TLS 1.3 / WSS	Plaintext WS	Imperative (FC1)
Lock actuator	12V Solenoid + TIP122	Servo motor	Fail-Secure + holding force
Cellular uplink	SIM800L (2G GPRS)	SIM7600 (4G LTE)	Cost + coverage in Algeria
GPS receiver	u-blox NEO-6M	NEO-M9N	Cost + Smart Mode fallback

Key insight

The technological stack of *Rafeeq* is the outcome of an explicit trade-off analysis along stable criteria. Every selected component is traceable both to a row of the requirements specification (Table 2.1) and to a comparison row of Table 2.10. This dual traceability is the analytical bridge between the functional requirements established in this chapter and the engineering deliverables of Chapter 3.

2.7 Conclusion

This chapter has formalised the engineering foundations of the *Rafeeq* system through the combined application of two complementary methodologies. The APTE functional analysis has established a structured requirements specification along six dimensions, identified eleven main and constraint functions, and traced each function to a technical solution through the FAST decomposition. The UML modelling has captured the behavioural dimensions of the system through four canonical diagrams that collectively describe what the system does, in which sequence, in which states, and over which data model.

The architectural decomposition has refined the three-layer model introduced in Chapter 1 into five interoperable subsystems — mobile application, backend cloud service, scooter onboard computer, docking station controller, and external payment integrations — each independently developable and deployable. The technological choices have been systematically

compared against credible alternatives along stable criteria, and every selection has been justified by reference both to the functional specification and to the operational constraints of the Algerian context.

Chapter 3 translates this architectural and functional foundation into the concrete engineering deliverables of the prototype. It details the design and in-house fabrication of the docking-station electronics, the BLE communication protocol implemented between the Raspberry Pi 3 Model B+ and the Xiaomi M365 ESC, the SIM800L 2G GPRS cellular uplink and the GPS-anchored anti-theft watchdog of the scooter subsystem, the captive-ring Solenoid locking mechanism of the docking station, the FastAPI backend with its REST and WebSocket interfaces, the Flutter mobile application across its sixteen screens, and the end-to-end latency and reliability measurements obtained on the integrated prototype.

CHAPTER 3

Design, Implementation and Validation of the Rafeeq System

3.1 Introduction

Chapters 1 and 2 established the contextual foundation and the functional architecture of the *Rafeeq* system. The first explained *why* a smart-mobility platform is needed in the Algerian First and Last Mile context; the second explained *what* such a platform should do and how it should be architected. The role of the present chapter is to explain *how* the platform was actually built, what it does in practice, and how it was verified to behave as intended.

The system that emerged from this engineering effort is composed of three physical objects and one cloud service. The three physical objects — a smart electric scooter equipped with embedded computing, an electromechanical docking station, and a mobile application running on the user's phone — are independent of each other in every respect: they live in different places, run on different hardware, are written in different programming languages, and have no way of speaking to each other directly. The cloud service — a FastAPI backend deployed on Render Cloud, backed by a PostgreSQL database — is the orchestrator that joins them together. Every command flows from one of the three physical objects, through the cloud, to one or more of the others; no other path exists. This deliberately centralised topology is what allows the system to remain coherent in the face of the heterogeneity of its parts, and what enables the central security model in which each physical object only needs to trust the cloud and not the others.

The presentation that follows is organised in the same order in which the three physical objects enter the user's experience, and ends with the cloud layer that binds them. After the high-level models of the system (Section 3.2), we describe the smart scooter and its IoT subsystem in full (Section 3.3), then dedicate Section 3.4 to the layered front-wheel lock and the GPS-anchored anti-theft system that protects the scooter when it is at rest. Section 3.5 then presents the docking station and the Solenoid lock, including the electronic driver circuit, the protection components, and the regulated power supply. Section 3.6 covers the Flutter mobile application across its sixteen functional screens. Section 3.7 introduces the cloud backend and explains in operational terms how it orchestrates the conversation between the three physical objects, including a formalisation of the IoT communication channels and the data flows

between them. Section 3.8 closes the design exposition by tracing a complete rental session step by step, from the moment the user taps a button on her phone until the Solenoid pin physically retracts. Sections 3.9 and onward then report on the in-house fabrication of the prototype circuit board, the experimental validation campaign, the strengths and limitations identified, and the conclusions of the work.

3.2 System Architecture and Modeling

3.2.1 Synoptic Diagram of the System

The synoptic diagram of Figure 3.1 presents the four functional blocks of the *Rafeeq* system and the principal energy and data exchanges between them. The user interacts with the system through the mobile application; the mobile application drives the cloud backend; the backend coordinates the two IoT endpoints (scooter and station); and the IoT endpoints close the physical loop by actuating the scooter lock and reading the scooter telemetry.

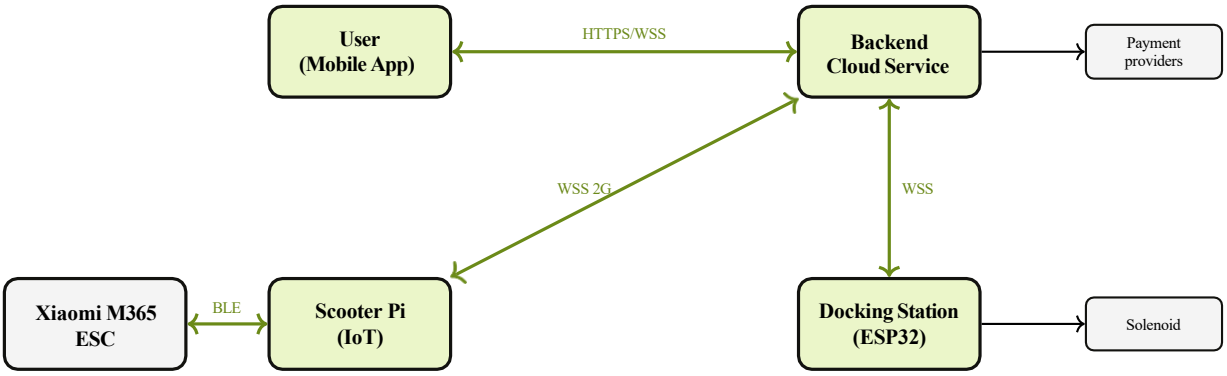


Figure 3.1. *Synoptic block diagram of the Rafeeq system showing the four functional blocks and the principal interfaces between them.*

3.2.2 Functional Flowchart

Figure 3.2 formalises the operational flow of a single rental session, from the moment the user opens the application to the final billing of the trip. The flowchart distinguishes synchronous user actions (rectangular blocks), system decisions (rhombi), and asynchronous events generated by the IoT layer (parallelograms).

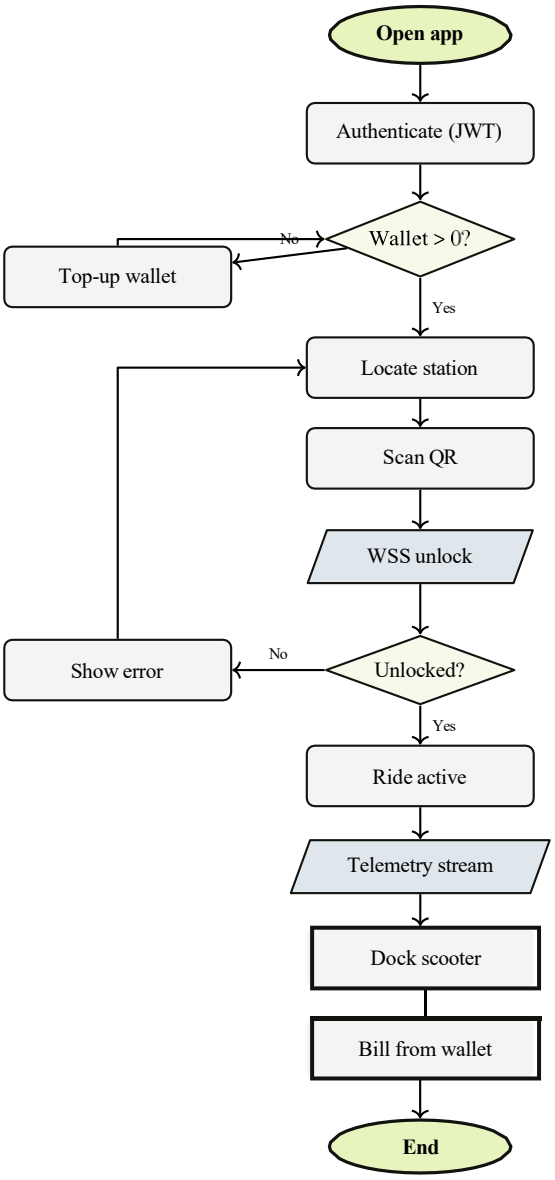


Figure 3.2. *Functional flowchart of a single Rafeeq rental session, from application launch to trip billing.*

3.2.3 Use Scenario

The canonical use scenario describes a typical interaction with the system from the user's perspective. A student leaves her residence to attend a class at the central campus, located 1.8 km away. She opens the *Rafeeq* application, identifies the nearest station on the map (approximately 80 metres from her residence), and walks there. Upon arrival, she scans the QR code printed on the scooter, the application confirms her wallet balance, and the backend issues the unlock command. The Raspberry Pi inside the scooter receives the command, transmits a BLE release command to the ESC, and the docking-station ESP32 simultaneously releases the mechanical Solenoid lock. Within three seconds of the QR scan, the scooter is physically detached and powered. The student rides to the campus, dismounts, walks her

scooter to the nearest *Rafeeq* station, and docks it in any free slot. The docking is detected by the station ESP32, the trip duration is finalised, the cost is debited from her wallet, and she receives an in-application receipt and a brief satisfaction question.

3.3 The Smart Scooter and IoT Subsystem

The scooter is the most challenging of the three independent objects in the system because it is the only one that has to be mobile, autonomous, and simultaneously connected to multiple radio links. It carries an onboard computer, a cellular modem, a GPS receiver, a Bluetooth Low Energy radio for the link to the vehicle's electronic speed controller, and the original lithium-ion traction battery of the Xiaomi M365 that powers all of the above. The four IoT modules together draw less than 4 W of continuous power, compared with roughly 250 W for the traction motor at cruising speed, so their contribution to the depletion of the traction battery is negligible. This section presents the complete hardware list, then describes each module in turn, and closes with the Python daemon that orchestrates them.

3.3.1 Complete Hardware List of the Scooter

Table 3.1. Complete hardware inventory of the Rafeeq smart scooter.

Component	Function
Raspberry Pi 3 Model B+ (1 GB)	Onboard computer running the Python daemon
u-blox NEO-6M GPS receiver	Position telemetry and anti-theft anchoring
SIM800L 2G GPRS modem	Cellular uplink to the cloud backend
XL4015 buck DC-DC converter	Step-down from 12 V auxiliary rail to 4.2 V SIM800L supply
470 µF bulk capacitor	Reservoir for SIM800L burst current draw
microSD card 32 GB Class 10	OS image and persistent state storage
Custom UART harness	Connects GPS and SIM800L to Pi UART pins
Xiaomi M365 electric scooter	Vehicle platform with built-in ESC, battery, and brakes

3.3.2 The Heart of the Scooter — Raspberry Pi 3 Model B+

The Pi 3 Model B+ is a single-board computer based on the Broadcom BCM2837B0 quad-core Cortex-A53 processor running at 1.4 GHz, with 1 GB of LPDDR2 SDRAM and 40 GPIO pins available for external interfacing. It runs Raspberry Pi OS Debian 13 (Bookworm) on a 32 GB microSD card, and all the *Rafeeq* application software running on it is written in Python 3.13. The choice of the Pi 3B+ over the more powerful Pi 5 was driven by three considerations. First, the power budget of the Pi 3B+ is approximately one third smaller, which extends the autonomy of the scooter under a fixed traction-battery capacity. Second, the existing Raspberry Pi OS Debian image used during the development of the daemon is fully backwards-compatible with the Pi 3B+ and migrating to the Pi 5 would have required

no functional change but would have introduced an avoidable revalidation effort. Third, the Pi 3B+ is more immediately available in the Algerian local market and at a lower unit cost than the Pi 5, which matters for any planned scale-up to several scooters.

The role of the Pi is central. It is the module that reads the position from the GPS, the module that opens the internet channel through the SIM800L, the module that talks to the scooter ESC over BLE to issue unlock, lock, and beep commands, and the module that runs the watchdog that monitors the position to detect theft attempts. All of these tasks run concurrently on a single core through the Python `asyncio` library, which allows multiple coroutines to share execution on a single thread by yielding control whenever they are waiting for I/O.

To guarantee that all of these services start automatically when the scooter is powered on, we rely on the `systemd` init system that ships with Raspberry Pi OS. Two `systemd` services are defined. The first, `rafeeq-ble.service`, runs with root privileges because the `bluepy` library that implements the BLE GATT client requires low-level access to the host controller interface. The second, `rafeeq-agent.service`, runs as an unprivileged user named `rafeeq` and is the service that contains all the application logic. The two services communicate through a Unix domain socket at `/tmp/ble_agent.sock`, with the agent service issuing BLE commands to the BLE service and receiving back the acknowledgement frames.

3.3.3 GPS Receiver — u-blox NEO-6M

Position fixes on the scooter are obtained from a GY-GPS6MV2 module based on the u-blox NEO-6M chipset, connected to an external ceramic patch antenna via a U.FL pigtail. The module outputs standard NMEA 0183 sentences on a UART interface at 9600 baud, with one fix per second under good sky conditions. A typical NMEA sentence carrying a position fix looks like this:

```
$GPGGA,143521.00,3604.39000,N,00445.66000,E,1,08,1.2,52.4,M,49.2,M,,*7B
```

The Pi reads these sentences through the `pyserial` library and decodes them with the `pynmea2` library, which exposes the latitude, longitude, altitude, speed, number of visible satellites, and horizontal dilution of precision as structured Python objects. The decoded fix is then bundled with M365 telemetry from the BLE link — battery state-of-charge, instantaneous speed, cumulative trip distance, ESC fault flags — into a JSON telemetry message that is transmitted to the backend every 30 seconds over the WebSocket connection. Each telemetry message has a typical payload size of 180 bytes; the resulting upload bandwidth during active sessions is approximately 200 bytes/s, well below the 40–80 kbps capacity of the 2G GPRS link.

Smart GPS Fallback Mode

The well-known limitation of low-cost GPS modules such as the NEO-6M is that they require a direct line of sight to the constellation, which they lose inside buildings or under dense urban canopies. To prevent the telemetry stream from going dark whenever the scooter enters such a covered area, the daemon implements a three-tier fallback policy that we call **Smart GPS Mode**. First, the daemon attempts to read a live fix from the hardware module; on success, the fix is forwarded to the backend tagged `live` and is also cached to disk in the file `gps_cache.json`. Second, if the hardware fails to deliver a valid fix within one second, the daemon reads the most recent cached fix from disk and forwards it tagged `cached`; the backend is then aware that the position is no longer in real time but is still recent. Third, if no cached fix exists — as on first boot after a fresh installation — the daemon falls back to a default coordinate read from the `.env` configuration file and tags it `demo`. This three-tier policy guarantees that the telemetry stream never carries a null position, which simplifies the backend logic significantly and removes the need to handle missing-position edge cases in the operator dashboard.

3.3.4 Cellular Uplink — SIM800L 2G GPRS Modem

The single hardest technical challenge in connecting a mobile scooter to the internet is the choice of the access technology. Wi-Fi is unsuitable because the scooter changes location continuously and no Wi-Fi network can be relied on to cover the route. Bluetooth tethering through a paired smartphone would tie the scooter to a particular user and is operationally unacceptable. The remaining option is the public cellular network, which has the great advantage of being available wherever an Algerian operator has coverage — in practice, in essentially every urban area where *Rafeeq* stations are likely to be deployed.

The SIM800L module was selected as the cellular modem because it is the cheapest available option that meets the requirement. It supports the 2G GPRS standard, which provides packet-switched data services over the GSM voice network. 2G is the oldest and slowest of the cellular generations, with a peak downlink throughput of 40–80 kbps on the standard CS-1 to CS-4 coding schemes; nonetheless, 2G covers approximately 95 % of the Algerian territory, including many regions where 3G and 4G are unavailable or unreliable. Given that the *Rafeeq* telemetry payload requires only 200 bytes/s of upload bandwidth, the 2G capacity is more than sufficient. The SIM800L communicates with the Pi over UART2 at 115200 baud and accepts a standard SIM card from any of the three Algerian operators (Mobilis, Djezzy, Ooredoo).

The principal electrical challenge with the SIM800L is its peak current draw of up to 2 A during burst transmission, which is well above what the Pi can deliver directly through its GPIO power rail. The supply path therefore includes a dedicated XL4015 buck DC-DC converter that steps the 12 V auxiliary rail down to the 4.2 V nominal supply voltage of the

SIM800L at up to 5 A, and a 470 µF bulk capacitor in parallel that absorbs the transient currents during the burst. With this provision in place, the SIM800L draws on average less than 100 mA in idle and the peak current is invisible to the rest of the system.

On boot, a Python script named `sim800l_manager.py` sends a sequence of AT commands to the modem to bring the GPRS data connection online. The sequence is the following:

1. `AT` — ping the modem; an `OK` response is expected.
2. `AT+CSQ` — query the signal strength; a value above 10 (out of 31) is considered acceptable.
3. `AT+CREG?` — check the network registration status; a response of `0,1` (home network) or `0,5` (roaming) is expected.
4. `AT+CGATT=1` — attach the device to the GPRS service.
5. `AT+CSTT="internet.djezzy.dz"` — set the access point name (APN) appropriate to the SIM operator.
6. `AT+CIICR` — bring up the wireless data link.
7. `AT+CIFSR` — query the assigned IP address.

After this sequence has executed successfully, the Pi can open arbitrary TCP/IP connections through the SIM800L exactly as if it were connected to a Wi-Fi network. The Python WebSocket client establishes a persistent WSS connection to the *Rafeeq* backend on top of this link, and the connection is maintained for the lifetime of the daemon, with automatic reconnection on any failure.

The recurring operating cost per scooter is dominated by the monthly subscription of the SIM card. With a low-volume data plan of 50 MB per month, which is more than two orders of magnitude above the actual data consumption of the telemetry, the monthly cost is approximately 200–300 DZD per scooter, which is significantly lower than the corresponding 4G/LTE option and is what makes the operating economics of the fleet sustainable.

3.3.5 Bluetooth Low Energy Link to the Xiaomi M365

The Pi 3 Model B+ contains a built-in Cypress CYW43438 wireless module that supports both Wi-Fi and Bluetooth Low Energy. In the *Rafeeq* scooter, the Wi-Fi capability is unused and the BLE radio is dedicated to communicating with the Xiaomi M365 electronic speed controller. The M365 ESC implements a proprietary BLE profile that exposes a command interface through a custom characteristic; the protocol has been reverse-engineered by the community and documented in technical reports [22, 30].

On the Pi side, the `rafeeq-ble.service` uses the `bluepy` Python library to open a GATT connection to the scooter by MAC address (in the prototype scooter, the address is

`EB:7D:B7:95:51:04`). Once connected, the daemon writes binary command frames to the command characteristic. Each frame follows the structure shown in Listing 3.3, with a two-byte header (`55 AA`), a length byte, source and destination identifiers, a one-byte command code, the command-specific payload, and a final checksum byte.

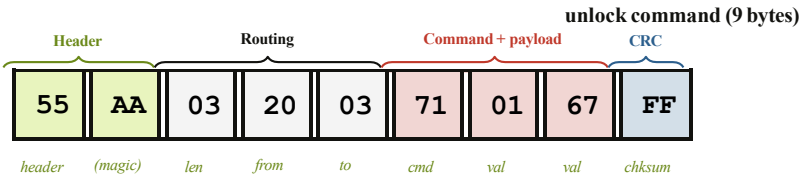


Figure 3.3. *Structure of the BLE binary command frame used by the Xiaomi M365 proprietary protocol, illustrated on the unlock command (`55 AA 03 20 03 71 01 67 FF`). The two-byte header magic identifies the protocol; the three routing bytes carry the frame length and the source/destination identifiers; the command code (`0x71` for unlock) is followed by its payload; the final byte is the CRC checksum.*

Upon receiving a valid frame, the M365 ESC verifies the checksum, executes the command, emits a short audible beep through its onboard piezo speaker, and either releases the electric brake on the front wheel (unlock) or engages it (lock). Table 3.2 summarises the principal BLE command codes used in the *Rafeeq* daemon.

Table 3.2. *Principal BLE commands sent by the Raspberry Pi to the Xiaomi M365 ESC.*

Command	Hex code	Effect
Unlock	`0x71`	Release the electric brake, enable throttle response
Lock	`0x70`	Engage the electric brake, disable throttle response
Get state	`0x20`	Read battery, speed, fault flags
Set speed limit	`0x73`	Configure speed limit (km/h)
Lights on/off	`0x40`	Toggle headlight
Beep	`0x41`	Audible feedback (acknowledgement, alert)

The `Beep` command in particular plays a key role in the anti-theft system described in the next section: it allows the daemon to make the scooter emit an audible alarm pattern on demand, without any unlock action and without depending on rotation of the wheel.

3.3.6 Onboard Enclosure

The four onboard modules — the Raspberry Pi, the SIM800L modem, the GPS receiver, and the XL4015 buck converter — are integrated inside a single 3D-printed enclosure mounted beneath the scooter deck. The enclosure is sealed against road spray to an IP54 equivalent and provides external connectors only for the BLE antenna, the GPS antenna patch, the SIM800L antenna, and the 36 V power input. Figure 3.4 shows the populated Raspberry Pi prototype and Figure 3.5 shows the assembled enclosure before installation on the scooter chassis.

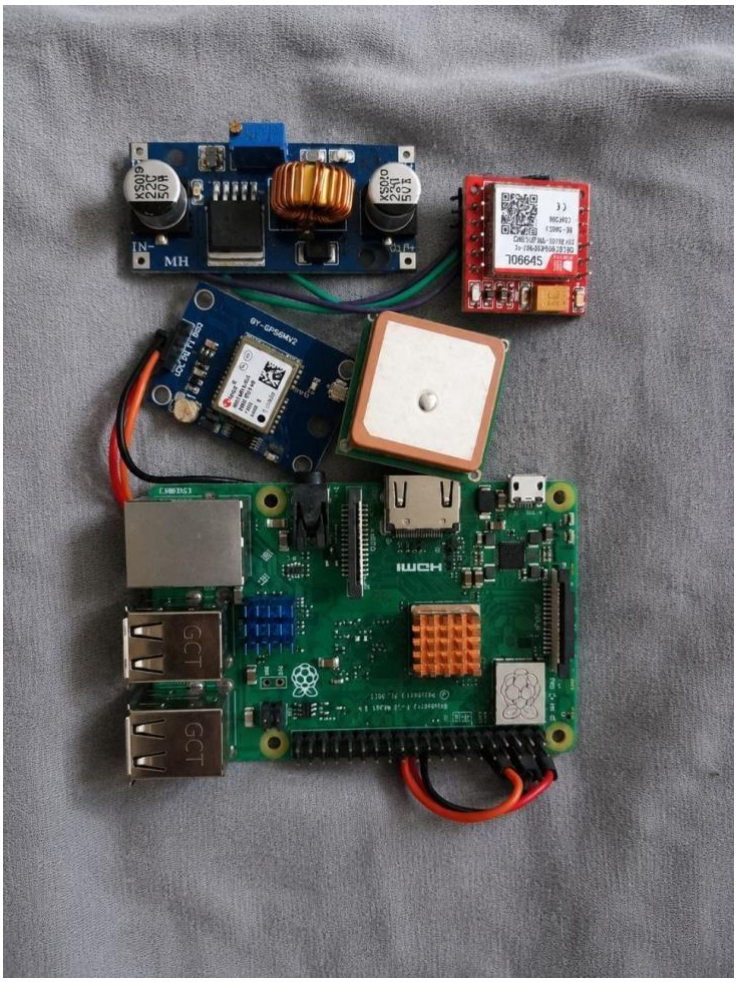


Figure 3.4. Onboard computer of the Rafeeq scooter prototype: Raspberry Pi 3 Model B+, SIM800L 2G GPRS modem, u-blox NEO-6M GPS, and the XL4015 buck converter that supplies the modem rail.

Figure 3.5. Assembled scooter IoT enclosure containing the Raspberry Pi 3B+, the SIM800L modem, the GPS receiver, and the buck converters, prior to mounting on the scooter chassis.

3.3.7 Python Daemon Architecture

The scooter daemon is implemented in Python 3.13 using `asyncio` for concurrency. The architecture is centred on three coroutines that run concurrently within a single event loop. The *BLE coroutine*, hosted by the `rafeeq-ble.service`, maintains the GATT link to

the M365 ESC, accepts commands from the agent service through the Unix socket, ex-ecutes them, and parses incoming telemetry notifications. The *WSS coroutine*, hosted by the `rafeeq-agent.service`, maintains the WebSocket Secure connection to the cloud backend through the SIM800L modem, sends telemetry messages every 30 seconds, and re-ceives backend commands as they arrive. The *controller coroutine*, also in the agent service, arbitrates between the two: it translates incoming backend commands into BLE command sequences forwarded through the Unix socket, and it aggregates BLE telemetry frames into the JSON payload sent over WSS. A fourth supervisor coroutine monitors the health of the three workers and restarts any that fail; the entire daemon process is itself supervised by `systemd` with automatic restart on crash, so the system recovers cleanly from any transient fault. Figure 3.6 shows a screenshot of the daemon logs during a successful BLE connection.

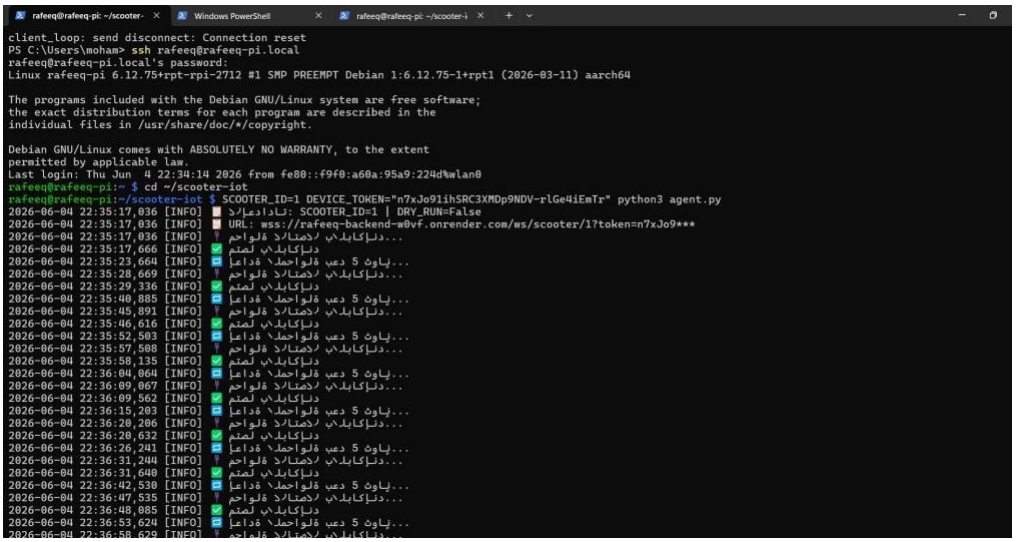


Figure 3.6. *Terminal output of the Python daemon during a successful BLE connection to the Xiaomi M365 ESC.*

3.4 Front-Wheel Lock and Anti-Theft System

Vehicle theft is the single largest non-collision cost driver of every micro-mobility platform operating today. International benchmarks of fleets in Europe and the United States report that between 5 % and 15 % of a deployed fleet is lost annually to theft and vandalism, with a strong concentration of incidents in the first months of operation when thieves probe the resilience of the security model. Any deployment in the Algerian context, where existing private bicycles and scooters are routinely stolen even from locked positions, must therefore treat anti-theft as a first-class engineering concern and not as an afterthought.

The *Rafeeq* prototype implements three layers of physical and digital protection, each addressing a different failure mode. The first layer is the original Xiaomi front-wheel elec-tronic brake. The second layer is the station-side captive ring that mechanically tethers the scooter to a fixed point. The third layer is a GPS-anchored anti-theft watchdog that detects

displacement above the position-fix noise floor and triggers both a local audible alarm and a backend alert. The combination of the three layers protects the scooter against the three distinct theft scenarios known from the field: pushing the scooter away on its wheels, prying it loose from the dock, and lifting it bodily off the ground.

3.4.1 Layer 1 — Front-Wheel Electronic Brake of the Xiaomi M365

The Xiaomi M365 ships with a built-in electronic lock that operates on the front wheel through its electronic speed controller. When the lock command is issued through the BLE link described in the previous section, the ESC engages the electric brake on the front motor and disables throttle response. In this state, the front wheel can still be pushed against the brake, but the resistance is large enough to make rolling the scooter on its two wheels impractical over any significant distance: the wheel turns reluctantly, with audible clicks from the electromagnetic detents of the brushless motor, and after a few metres the rider gives up.

Importantly, the ESC also implements an autonomous audible warning. If the front wheel is forced to rotate beyond a small angular threshold while the lock is engaged, the ESC emits a sequence of warning beeps through its onboard piezo speaker — approximately five beeps per second, audible up to 10–15 metres away — alerting any bystander to the unauthorised movement. The warning sequence is generated locally by the ESC firmware and does not depend on any external system; it works even if the scooter is fully off-grid.

The *Rafeeq* daemon issues this front-wheel lock command via BLE every time a rental session ends, immediately before disconnecting the BLE link, so that the scooter enters its passive state already locked. The lock state persists across power cycles, so even if the M365 traction battery is removed and reinserted, the wheel remains locked until a legitimate unlock command is received over BLE.

3.4.2 Layer 2 — Captive Ring at the Docking Station

The second layer is the captive steel ring that is permanently attached to the scooter frame and is mechanically captured by the docking-station Solenoid pin when the scooter is docked. The principle of operation of this mechanism is described in detail in Section 3.5: in summary, the pin passes through the ring while the scooter is docked, holding it against any pulling, lifting, or torsional force, and is retracted only on receipt of an explicit unlock command from the backend. Even if a determined thief manages to bypass the front-wheel lock by carrying the scooter off the ground, the captive ring keeps the scooter tethered to the station until the backend issues the unlock command. The pin and the ring are both manufactured in hardened steel, so the failure mode is the destruction of the polymer station body rather than the failure of the lock itself.

3.4.3 Layer 3 — GPS-Anchored Anti-Theft Watchdog

The third layer protects the scooter specifically against the lifting-and-carrying scenario, which is the failure mode that the first two layers cannot detect. A thief who lifts the entire scooter into a truck or onto a hand-cart defeats both the front-wheel brake (which depends on the wheel rotating against resistance) and the captive ring (which only protects against detachment from the docking pin). What no determined thief can defeat without rebooting or destroying the Pi is a watchdog that simply observes the position of the scooter and reacts to any displacement above the noise floor.

The watchdog operates as follows. At the moment a rental ends and the scooter enters its locked state, the daemon captures the current GPS position and stores it in memory as an **anchor point**. From that moment until the next legitimate unlock event, the daemon evaluates the great-circle distance from the anchor to every new GPS fix using the Haversine formula:

$$d = 2R \arcsin\left(\sqrt{\sin^2(\Delta\varphi 2) + \cos\varphi_1 \cos\varphi_2 \sin^2(\Delta\lambda 2)}\right) \tag{3.1}$$

where $R = 6{,}371{,}000$ m is the Earth's mean radius, φ_1, φ_2 are the latitudes of the anchor and the current fix expressed in radians, and $\Delta\varphi$, $\Delta\lambda$ are the latitude and longitude differ-ences also in radians. The Python implementation of the formula in the daemon is a direct transliteration of Equation eq:haversine and consumes a negligible fraction of CPU time per evaluation.

If the computed distance d is less than 15 metres, the daemon ignores the fix, because a distance below the GPS noise floor is consistent with the scooter sitting still at the dock. The threshold of 15 metres was chosen empirically to be approximately three times the typical 2.5–5 m standard deviation of consecutive NEO-6M fixes under good-sky conditions, which yields a false-positive rate below one event per day in the prototype dataset. If the computed distance exceeds 15 metres, the daemon classifies the event as suspected theft and triggers the dual response described below.

Dual Response — Local Alarm and Backend Alert

The dual response runs two parallel actions in the same coroutine of the daemon, on the millisecond after the threshold is crossed.

Action one — audible alarm on the scooter. The daemon issues a sustained BLE `Beep` command to the M365 ESC, configured to produce a high-pitched alarm pattern of five seconds of continuous tone followed by two seconds of silence, repeating indefinitely. This pattern is intentionally distinct from the original Xiaomi short-beep pattern used for the rotation warning, so a passer-by can recognise it as an active alarm rather than a routine warning. The pattern has two functions. First, it signals to the thief that the scooter is being actively

monitored and is not a casually abandoned object; the social cost of carrying off an alarm-emitting object in a public space is high enough to deter many would-be thieves. Second, it draws the attention of bystanders, who in the Algerian urban context routinely intervene or alert the police when an alarm sounds.

Action two — backend alert. In the same instant, the daemon emits a structured JSON event over the WebSocket link to the backend, containing the scooter identifier, the anchor coordinates, the current coordinates, the computed distance in metres, the GPS timestamp, and a sequence number. The backend persists the event in the `theft_alerts` table of the PostgreSQL database, displays the event in real time on the operator dashboard alongside a map of the displacement vector, and emits a push notification to the operator's mobile phone. The operator can then dispatch the field team to recover the scooter; the displacement vector continues to update on the dashboard as long as the GPS link is alive, so the field team always sees the latest position.

Termination Conditions of the Alarm

The alarm pattern stops automatically when any one of three conditions is satisfied. First, if the scooter returns within 15 metres of the anchor — for instance because the thief drops it and the original user picks it up — the daemon issues a BLE command to silence the speaker and reverts to the silent watchdog state. Second, if a legitimate unlock command is received from the backend — because the original user starts a new rental session on the same scooter — the alarm is silenced as part of the unlock procedure. Third, if the operator explicitly cancels the alert from the dashboard — because she has determined that the displacement is legitimate, for instance during a maintenance pickup or a manual fleet rebalancing operation — the backend pushes a cancellation event to the daemon, which silences the alarm.

Why This Layered Architecture Works

The critical property of this anti-theft architecture is that the three layers cover non-overlapping failure modes. The front-wheel brake addresses pushing-and-rolling theft; the captive ring addresses prying-off-the-dock theft; the GPS watchdog addresses lifting-and-carrying theft. No single piece of hardware can defeat all three layers simultaneously without leaving an evidence trail that an operator can use to track and recover the scooter. The marginal hardware cost of the third layer is zero, because the GPS receiver, the BLE radio, and the WSS uplink are all already deployed for other purposes; the only addition is a few hundred lines of Python code that connect the watchdog logic to the existing infrastructure.

3.5 The Docking Station and Solenoid Lock

If the scooter is the most challenging of the three independent objects, the docking station is the simplest. Its job is narrow and well-defined: receive an unlock or a lock command from the backend, and either release or capture the scooter in response. Everything that the station does — the electronics, the firmware, the mechanical lock, the physical enclosure, the user-facing QR plate — is in the service of that single function. This narrowness of purpose is a feature, not a limitation: it allows the station to be inexpensive, low-power, and easy to fabricate in quantity for an eventual fleet rollout.

This section presents the station in five layers, moving from the complete hardware inventory through the heart of the controller, the lock mechanism, the driver circuit, the protection components, and the power supply.

3.5.1 Complete Hardware List of the Station

Table 3.3. Complete hardware inventory of the Rafeeq docking station.

Component	Function
ESP32 Dev Module	Station controller with built-in Wi-Fi
Solenoid Lock 12 V (Electric Bolt)	Physical lock actuator
TIP122 Darlington NPN Transistor	High-current electronic switch for the Solenoid
1N4007 Diode	Flyback protection against the inductive back-EMF
LM7805 Regulator	Steps the 11.1 V battery down to a regulated 5 V for the ESP32
220 µF Electrolytic Capacitor	Reservoir on the regulated 5 V rail
1 kΩ Resistor	Base current limiting on the TIP122
10 kΩ Resistor	Pull-down on GPIO 13 to suppress boot-time floating
3× 18650 Li-ion cells in series	11.1 V nominal power source
J1 Connector	External terminal for the Solenoid output

3.5.2 The Heart of the Station — ESP32

The ESP32 is a low-cost dual-core microcontroller from Espressif Systems, built around the Xtensa LX6 architecture and running at up to 240 MHz, with 520 KB of SRAM, integrated Wi-Fi 802.11 b/g/n, and Bluetooth 4.2 plus BLE [31]. Its retail price in the local market is below 1000 DZD, which is the principal reason it was selected for the station controller: a fleet rollout with hundreds of stations is economically sensitive to the unit cost of each station, and the ESP32 brings full networking capability at a fraction of the cost of any other comparable controller.

The ESP32 firmware is written in C++ in the Arduino IDE, with a total source size of approximately 200 lines that nevertheless implements the full WebSocket protocol, the JSON parser for incoming commands, the GPIO drive of the Solenoid, and the state machine that

arbitrates between them. On power-up, the `setup()` function initialises the Serial port for diagnostic logging, configures GPIO 13 as a digital output with its default value set to `LOW` (so that the lock starts in the engaged state under all power-on conditions), and connects to the configured Wi-Fi network. Once Wi-Fi is up, the ESP32 opens a secure WebSocket connection (WSS) to the cloud backend at the URL `wss://rafeeq-backend-w0vf.onrender.com/ws/` where the `1` is the station identifier and the `token` is the 32-byte `device_token` generated for this specific station when it was provisioned in the database. The backend verifies the token against the value stored for the station, accepts the connection, and updates the `is_lock_online` flag of the station record to `true`. From this moment, the station is ready to receive commands.

3.5.3 The Solenoid Lock — 12 V Electric Bolt with Captive Ring

The actual locking is performed by a 12 V electromechanical solenoid of the Electric Bolt type, mechanically coupled to a steel pin (*lisan*) housed inside the station body, combined with a hardened-steel ring (*captive ring*) that is permanently attached to the scooter frame. The principle of operation is the inverse of a conventional sliding bolt: the scooter does not engage the lock directly, but rather through the ring that it carries with it at all times.

When a scooter is docked, the user inserts the ring into a slot machined in the side of the station; the pin — held in the extended position by an internal compression spring — passes through the ring and physically captures it, immobilising the scooter against any pulling, lifting, or torsional attempt. This is the *locked* state, and it is the natural rest state of the system. In the absence of any electrical command, and even in the event of a total power outage, the spring keeps the pin extended through the ring and the scooter remains secured. Such a lock is therefore called *Fail-Secure*: any failure of the controller, the network, or the power supply leaves the scooter locked rather than free.

When the user requests an unlock through the mobile application, the backend transmits an unlock command over the WebSocket gateway, the ESP32 receives the command, parses the JSON payload, and drives GPIO 23 high (in the prototype, the lock-control GPIO was assigned to pin 13 of the dev-module pinout; the schematic in Figure 3.8 uses the symbolic name GPIO 23 for the same physical signal). The 3.3 V signal saturates the base of the TIP122 transistor of the driver circuit (Section 3.5.4), 12 V is applied to the Solenoid coil, the magnetic field overcomes the spring force, and the pin retracts inside the station body within approximately 200 ms. The user then removes the ring from the slot — and, with it, the scooter — and the rental session is operationally active.

After a configurable activation window of 500 ms, the ESP32 returns GPIO 13 to `LOW`: the Solenoid de-energises, and the spring returns the pin to the extended position, ready to capture the next ring that is inserted by a returning user. The holding force generated by the spring at the engaged position is approximately 70 N, which is sufficient to resist any

reasonable manual attempt to extract the ring without an unlock command. The Solenoid specifications are summarised in Table 3.4.

Table 3.4. *Specifications of the 12 V Solenoid Electric Bolt used in the prototype.*

Parameter	Value
Operating voltage	12 V DC
Current draw (engaged)	500 mA
Current draw (de-energised)	0 mA
Pin holding force at rest	≈ 70 N
Pin retraction time (under nominal supply)	≈ 200 ms
Failure mode	Fail-Secure (locked)
Body material	Hardened steel

Figure 3.7 shows the assembled lock subsystem of the prototype, with the electronic board on top, the Solenoid and pin on the bottom-left, and the captive ring on the bottom-right.

Figure 3.7. *Complete locking subsystem of the Rafeeq docking station, showing the electronic control board (top), the 12 V electromechanical Solenoid that drives the steel pin housed inside the station body (bottom-left), and the captive steel ring permanently attached to the scooter frame that the pin captures when the scooter is docked (bottom-right).*

3.5.4 The Driver Circuit — TIP122 Darlington Switch

The principal electrical challenge of the station is that the ESP32 GPIO can source only 40 mA at 3.3 V, while the Solenoid coil draws 500 mA at 12 V. There is no way to drive the Solenoid directly from a GPIO pin: a power-stage transistor is mandatory between the two.

The transistor selected is the TIP122, a Darlington NPN with a 100 V collector-emitter rating and a 5 A continuous collector current capability [24]. The Darlington topology —

two NPN transistors connected internally so that the emitter of the input transistor drives the base of the output transistor — gives the device a typical DC current gain h_{FE} of 1000 at a collector current of 3 A. Practically, this means that a base current of 0.5 mA from the ESP32 GPIO is sufficient to drive the 500 mA collector current required by the Solenoid, with a 1000-fold safety margin on the GPIO drive capability.

The connection topology is the following: GPIO 13 of the ESP32 drives the base of the TIP122 through a current-limiting resistor of 1 kΩ, which both protects the GPIO against accidental shorts and sets the base current to a safe operating point. The collector of the TIP122 is connected to one terminal of the Solenoid coil; the other terminal of the coil is connected to the 12 V battery rail. The emitter of the TIP122 is connected to the common ground rail of the station.

When GPIO 13 transitions from `LOW` to `HIGH`, the base sees 3.3 V minus the cumulative V_{BE} drop of the Darlington (approximately 1.4 V), the base current flows, the transistor saturates, the collector-emitter voltage drops to less than 1 V, and the full 12 V minus the saturation drop is applied across the Solenoid coil. The pin retracts; the scooter is released. When GPIO 13 returns to `LOW`, the base current is removed, the transistor cuts off, and no current flows through the coil. The spring returns the pin to the extended position; the next ring inserted will be captured.

The complete electronic schematic of the Rafeeq docking-station control circuit, designed using Proteus ISIS, is presented in Figure 3.8.

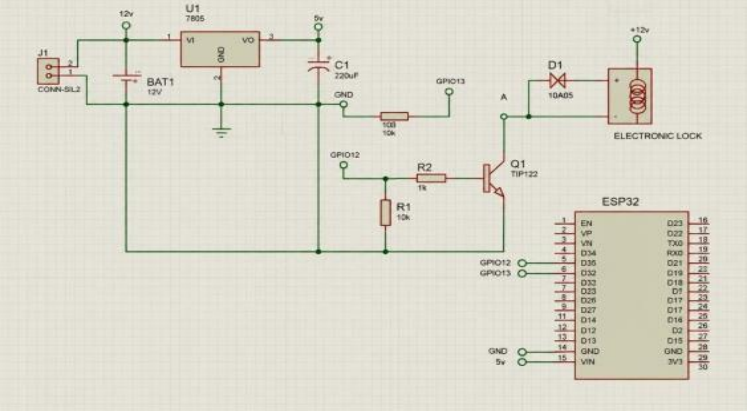


Figure 3.8. *Complete electronic schematic of the Rafeeq docking-station control circuit designed in Proteus ISIS.*

3.5.5 Flyback Protection — 1N4007 Diode

A Solenoid coil is an inductive load and stores energy in its magnetic field during operation. When the TIP122 switches off, the current through the coil cannot fall instantaneously; the inductor instead generates a high-voltage transient in the opposite polarity — the so-called *back electromotive force* or back-EMF — that can reach 100 V or more in the first microseconds after switch-off. This transient, if left unmanaged, would either destroy the TIP122 by exceeding its collector-emitter rating, or destroy the ESP32 if the transient finds a path back to the GPIO.

The standard solution is a *flyback diode* (also called *freewheel diode*), placed in paral-lel with the Solenoid coil with its cathode toward the positive rail and its anode toward the transistor collector. During normal operation, the diode is reverse-biased by the 12 V across the coil and carries no current. At switch-off, the polarity of the voltage across the coil reverses, the diode becomes forward-biased, and the stored magnetic energy circulates through the closed loop formed by the coil and the diode, dissipating as heat in the diode's bulk resistance over a few milliseconds.

The 1N4007 was selected for this role because of its 1 A average forward current rating (well in excess of the 500 mA Solenoid peak), its 1000 V reverse breakdown (far above

expected transient), and its mature supply chain that makes it available everywhere at a unit cost below 5 DZD [34].

3.5.6 Regulated Power Supply — LM7805 with Bulk Capacitor and Pull-Down

The ESP32 module requires a regulated 5 V supply on its VIN pin. The station prototype is powered by three 18650 lithium-ion cells connected in series, giving a nominal 11.1 V (and up to 12.6 V when fully charged). The LM7805 three-terminal linear voltage regulator steps this voltage down to a stable 5 V with a typical dropout voltage of 2 V and a maximum output current of 1 A [35], which is comfortably above the ESP32 quiescent draw of approximately 80 mA. At this load the LM7805 dissipates approximately $(12 - 5) \times 0.08 = 0.56$ W, which is within the thermal envelope of the TO-220 package operating in free air without a heat sink.

The choice of a linear regulator over a switching regulator for the ESP32 supply is deliberate. A switching regulator at typical 100 kHz to 1 MHz frequencies generates electromagnetic noise that can couple into the Wi-Fi receiver of the ESP32 and degrade its sensitivity; a linear regulator generates no such noise and produces a clean rail at the cost of a few hundred milliwatts of efficiency loss. For a station with a steady 80 mA draw, the efficiency loss is operationally negligible.

Two passive components surround the LM7805 to ensure correct behaviour under the transient current draw of the Wi-Fi radio. The first is a 220 µF electrolytic capacitor connected between the LM7805 output and ground, which acts as a reservoir during the few-milliseconds bursts of current that the ESP32 demands when it transmits a Wi-Fi frame. Without this capacitor, the LM7805 output would briefly dip below 4 V on each transmission, the ESP32 would enter brownout reset, and the WSS link would never come up reliably. The second is a 10 kΩ pull-down resistor connected between GPIO 13 and ground. Its role is to prevent the GPIO from drifting to an indeterminate level during the milliseconds of the ESP32 boot sequence before the firmware has had a chance to configure the pin as a low-driven output. Without this pull-down, the floating GPIO can settle at an intermediate voltage sufficient to partially saturate the TIP122 base, which would inadvertently energise the Solenoid and unlock the scooter at every boot. With the pull-down in place, the GPIO is firmly held at ground during the boot window and the Solenoid stays in its locked rest state until the firmware explicitly drives it.

3.5.7 Operating Principle and Firmware State Machine

The station firmware implements a finite-state machine with four states: *idle* (no scooter detected, lock engaged), *occupied* (scooter detected and locked), *releasing* (unlock command received, Solenoid energised), and *released* (scooter detected as removed, ready for the next

docking event). State transitions are triggered by combinations of incoming WebSocket messages, dock-occupancy sensor readings, and internal timers. The state machine is reset to *idle* on any unexpected disconnection or error, and any state change is immediately reported to the backend. Between commands, the ESP32 sends a ping frame every 20 seconds to keep the WSS link alive through any intermediate NAT timeouts; the backend replies with a pong frame, and if four consecutive pings go unanswered the ESP32 disconnects and reconnects automatically.

3.5.8 Station Enclosure and User Interaction

The control board, the Solenoid, and the battery are housed in a sealed compartment located at the base of the station, accessible for maintenance through a screwed cover protected by a tamper-evident seal. Cable entries are provided through gland fittings rated IP54. Each scooter slot in the station carries a printed QR code containing the scooter's identifier and a deep-link URL; the QR code is mounted at a height of approximately 1.1 m from the ground to allow ergonomic scanning by users of varying heights and is printed on a polycarbonate plate protected against UV degradation and abrasion. Figure 3.9 shows the assembled station prototype with a docked scooter, and Figure 3.10 shows the populated electronic control board on its own.

Figure 3.9. *Assembled prototype of the Rafeeq docking station with a scooter docked into the slot. The mechanical capture is implemented by the captive ring permanently attached to the scooter frame, which engages the Solenoid pin housed in the station body.*

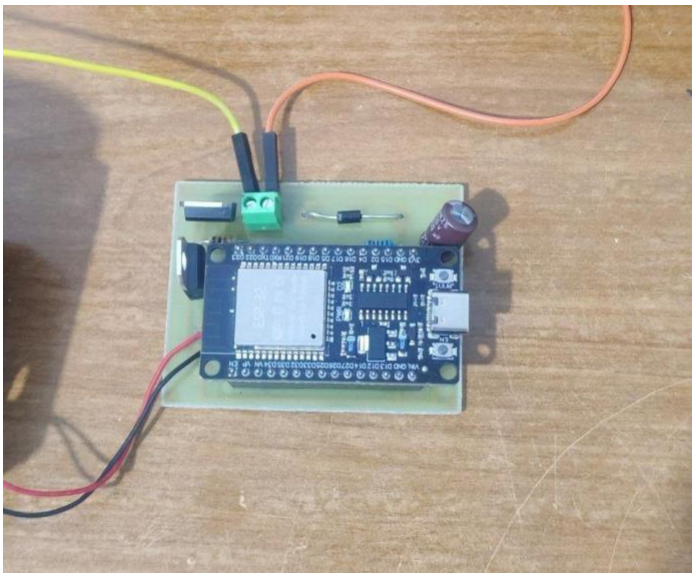


Figure 3.10. Prototype electronic board of the docking station with the ESP32, TIP122 Darlington driver, LM7805 regulator, 220 µF bulk capacitor, 1N4007 flyback diode, base and pull-down resistors, and the green screw-terminal output for the Solenoid coil.

3.6 The Mobile Application

The mobile application is the only piece of the *Rafeeq* system that the user ever touches directly. Every action she takes — creating an account, finding a station, scanning a QR code, paying for a ride, reporting a problem — passes through it. The design therefore had to balance two competing requirements. On the user-facing side, the application had to be fast, visually polished, and immediately understandable on a first encounter, since the entire value proposition of the service is to be quicker than the alternative of waiting for a bus. On the engineering side, the application had to talk to the same backend that talks to the IoT endpoints, share the same JWT authentication model, and react in real time to the events that the backend pushes whenever a scooter state changes. The two requirements meet in a cross-platform Flutter codebase organised around sixteen functional screens.

3.6.1 Framework Choice — Flutter and Dart

The application is a Flutter project written in Dart 3.4 and targeting Flutter 3.22. Flutter was selected over the native alternatives (Kotlin for Android, Swift for iOS) because it allows a single codebase to compile to Android, iOS, and the web with no platform-specific branches in the common business logic. For a small team without the resources to staff two parallel native projects, this is a decisive cost advantage and aligns the engineering capacity with the available human resources. The Dart language is itself a Google product, syntactically close to Java and JavaScript, with a static type system and a sound null-safety model that catches a substantial class of runtime errors at compile time. The application source code currently

totals approximately 4000 lines of Dart, distributed across the feature folders described in the next subsection.

3.6.2 Architecture — Clean Architecture in Three Layers

The internal architecture follows the Clean Architecture pattern, which separates the application into three layers each of which has a single responsibility and depends only on the layer below it [26, 27]. The **presentation** layer holds the screens and the visual widgets and knows nothing about the network or the database; its only role is to render the data that the layer below it provides, and to forward user events upward. The **application** layer holds the state managed by Riverpod providers, which receive events from the presentation layer, request data from the data layer, transform it as needed, and re-expose it to the presentation layer in a form ready for rendering. The **data** layer holds the API clients and the storage abstractions; it speaks HTTP through the `dio` library and returns plain Dart objects that the rest of the application understands. This three-layer separation is what allows the codebase to remain testable and maintainable as it grows; each layer can be modified independently of the others provided its contract with the adjacent layers is preserved.

Each feature of the application — authentication, wallet, ride, profile — is grouped in its own folder containing its own presentation, application, and data subfolders. Navigation between the screens of different features is declarative through GoRouter, with type-safe route definitions and deep-link support for QR code unlocking. The HTTP client is Dio, configured with interceptors that automatically attach the JWT access token and transparently refresh it when it expires. The JWT is stored in the platform-specific secure storage — Android Keystore on Android, Keychain on iOS — through the `flutter_secure_storage` package.

3.6.3 The Sixteen Screens

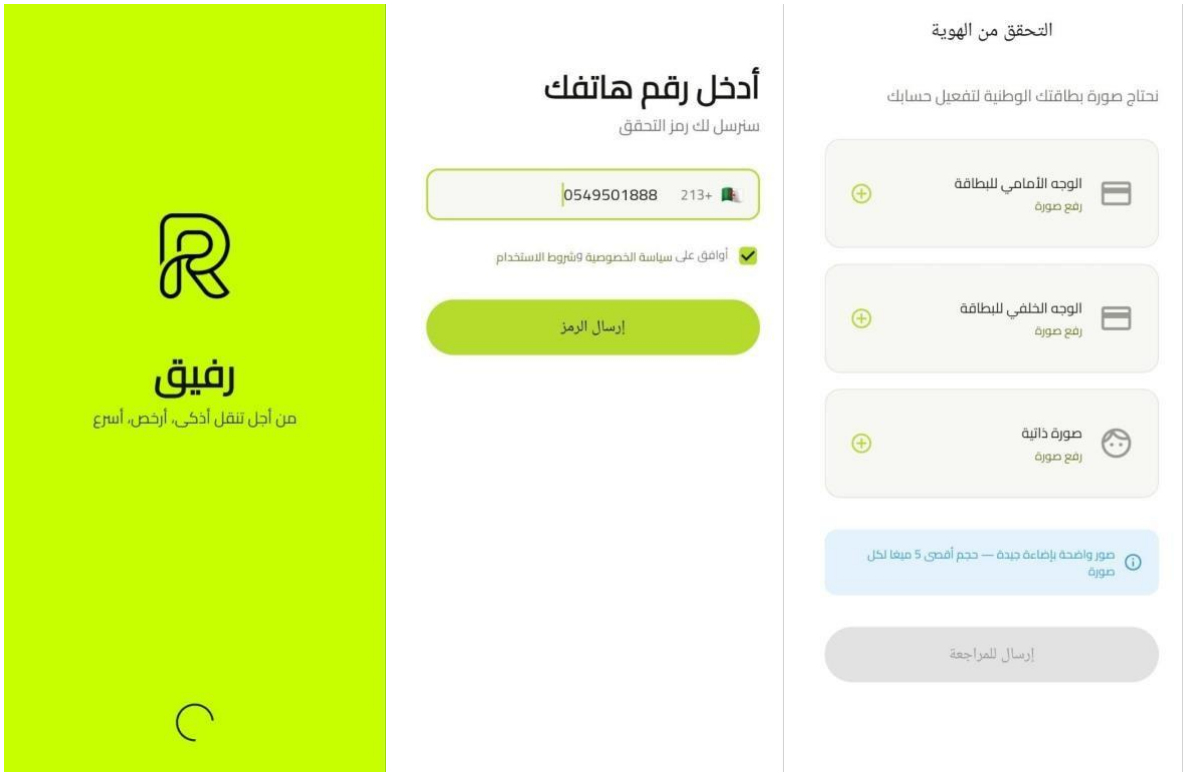
The application covers sixteen functional screens, each implementing one elementary step of the user journey. Table 3.5 lists the screens grouped by functional area.

Table 3.5. *Functional screens of the Rafeeq mobile application.*

Group	Screens
Onboarding	Splash, Onboarding (3 pages), Language picker
Authentication	Phone entry, OTP verification, KYC document upload
Discovery	Home / Map, Station detail, Scooter detail
Ride	QR scanner, Active ride, Trip summary
Wallet	Wallet balance, Top-up, Transaction history
Profile	Profile, Settings, Help & report

Authentication and Onboarding Flow

The authentication flow opens with a Splash screen that hides the start-up time of the application by displaying the brand mark, followed by three onboarding pages that summarise the value proposition of the service for first-time users. The phone-entry screen prefills the country code `+213` and asks the user to enter her Algerian mobile number; the backend triggers an OTP delivery to that number, and the user enters the six-digit code on the next screen. Upon successful OTP verification, new users are routed to the KYC document upload screen, where they photograph their identity document and a selfie; existing users are routed directly to the home map. The full onboarding completes in approximately 78 seconds for a first-time user, well below the 90-second target established in Chapter 2. Figure 3.11 illustrates the three principal screens of the authentication flow.



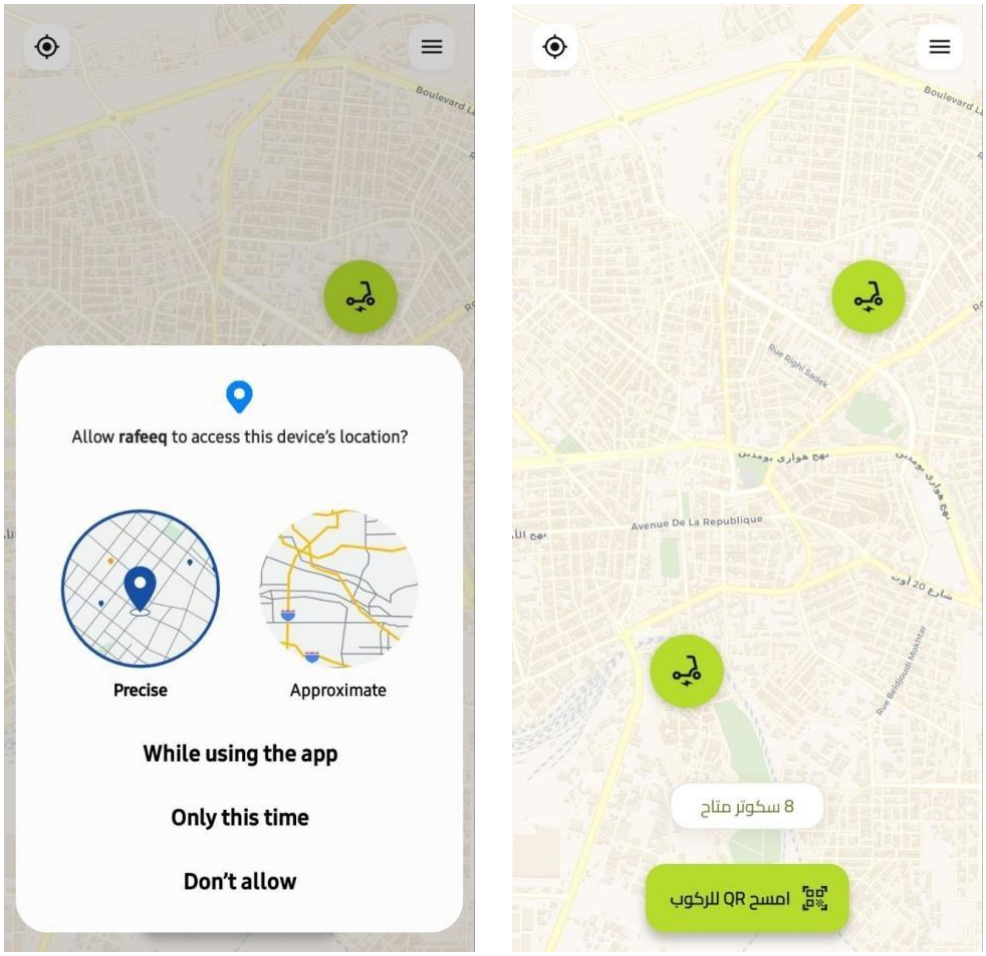
(a) Splash *(b) Phone entry* *(c) OTP verification*

Figure 3.11. *Authentication and onboarding flow of the Rafeeq mobile application.*

Map and Station Discovery

The home screen presents a map centred on the user's current location, with all nearby *Rafeeq* stations rendered as markers (Figure 3.12). The user's location is obtained through the platform geolocation API with permission requested on first launch. Station markers indicate the number of available scooters and are colour-coded by occupancy level; tapping a marker opens the station detail bottom sheet, which displays the name of the station, the count of available scooters, the list of scooter identifiers, the walking distance and time, and a direc-

tions shortcut to the platform map application. The map tile layer is OpenStreetMap, served through a CORS-proxied tile server.

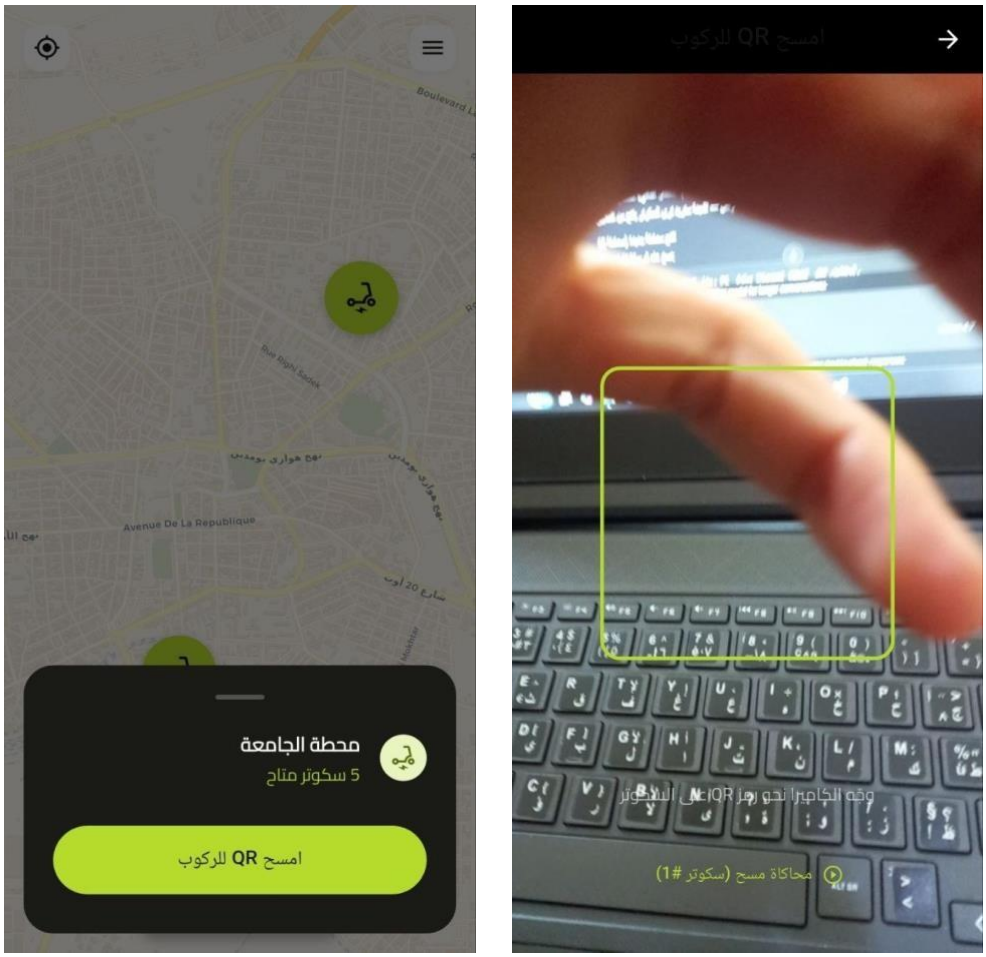


(a) Home map with station markers *(b) Station detail bottom sheet*

Figure 3.12. *Map and station detail screens of the mobile application.*

QR Unlock and Active Ride

The QR scanner is launched either from the floating action button on the home screen or from the scooter detail sheet (Figure 3.13). The scanner uses the `mobile_scanner` package, which wraps the platform-native camera APIs and decodes QR codes locally on the device. Upon successful scan, the scooter identifier is extracted from the QR payload, the application transmits an unlock request to the backend, and a transient progress overlay is shown until the backend confirms the unlock or returns an error. The total user-perceived latency from QR scan to unlock confirmation averages 2.83 seconds in field measurements. During the active ride that follows, the application displays a full-screen view with the elapsed time, the live cost computed from the per-minute tariff, the scooter battery percentage, the instantaneous speed, and a pause button. The data are refreshed every 500 ms through the WebSocket telemetry stream. The pause button suspends billing for up to ten minutes, allowing users to perform short stops without ending the trip.

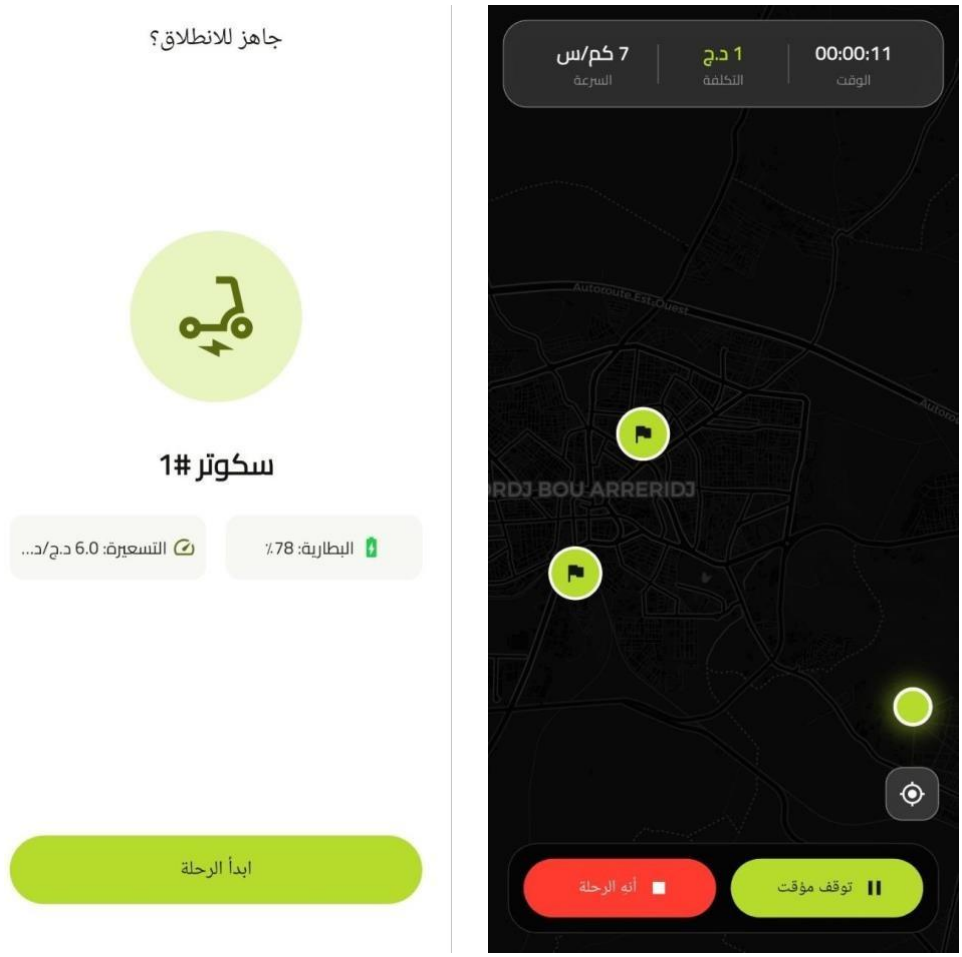


(a) QR scanner camera view *(b)* Active ride dashboard

Figure 3.13. *QR scanner and active ride interface of the mobile application.*

Wallet and Payment Channels

The wallet screen displays the current balance in Algerian Dinars and the recent transaction history (Figure 3.14). Top-up is performed through one of three channels: a bank card (CIB or Dahabia) processed through a SATIM-compatible gateway, BaridiMob through a deep-link integration, or a cash recharge at a partner retail outlet through a one-time code printed on the user's screen. All top-up channels redirect the user to a confirmation screen, and the wallet balance is updated in real time through a WebSocket push from the backend.

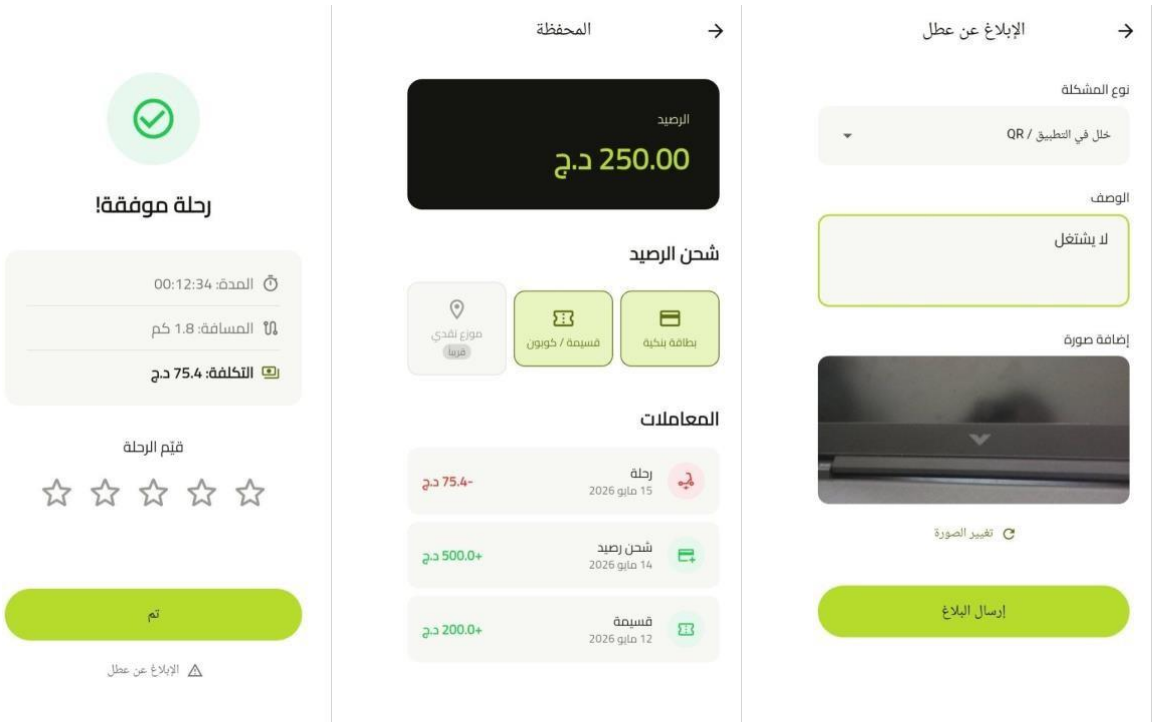


(a) Wallet balance *(b) Top-up channel selection*

Figure 3.14. *Wallet balance and top-up flow of the mobile application.*

Trip End, History, and Profile

After a successful trip end, the application displays a trip summary screen with the duration, the distance, the cost, the CO_2 saved relative to a private car, and a satisfaction question (Figure 3.15). The satisfaction question is a single five-star scale with an optional comment field; the data is transmitted to the backend and aggregated into operator dashboards. The trip history screen lists the user's past trips in reverse chronological order with a tap-to-expand summary, and the profile screen surfaces the user's lifetime ride count, the total distance travelled, and a CO_2 avoidance indicator that gamifies the environmental dimension of the service.



(a) Trip summary *(b) Trip history* *(c) Profile and settings*

Figure 3.15. *Trip summary, history, and profile screens of the mobile application.*

3.6.4 User-Experience Details

The default language of the application is Arabic with right-to-left layout, with English and French as alternative localisations selectable through the language picker. The principal Arabic font is Cairo and the principal Latin font is Inter. The primary colour palette is a combination of the brand VoltGreen — evocative of the electrical nature of the vehicle — and a neutral charcoal grey for the body text and the chrome of the user interface.

All inter-screen transitions use a smooth animation curve at 250 ms, which is the perceptual threshold above which the transition becomes noticeable and below which it remains imperceptible. The QR scanner action triggers a distinctive entrance animation that draws the user's attention to the camera frame, and the successful trip end is celebrated by a short confetti animation that rewards the user visually. These small attentions are what distinguish a polished application from a merely functional one.

JWT tokens are stored in `flutter_secure_storage`, which uses the platform-native Keychain on iOS and the EncryptedSharedPreferences on Android. No wallet balance or other sensitive data is cached locally; every piece of information that the user sees on the screen is fetched fresh from the backend on demand, which guarantees that the displayed state is always consistent with the source of truth on the server.

3.7 The Cloud Backend and Connection Layer

The three pieces presented so far — the scooter, the station, the mobile application — live in three different physical worlds and have no way of speaking to each other directly. The mobile application is in the user's pocket, possibly several kilometres away from the scooter. The scooter is on the street, talking to a 2G cellular tower. The station is bolted to a pole, talking to a Wi-Fi access point. The role of the backend is to be the only piece of the system that all three of them can reach, and to translate between their languages so that they appear, from each other's point of view, as a single coherent system. Every command travels from one of the three pieces, through the backend, to one or more of the other two; no other path exists.

3.7.1 The Backend Service — FastAPI on Render Cloud

The backend is implemented as a FastAPI application running on Python 3.13. The service is organised around a domain-driven layered architecture summarised in Figure 3.16. The HTTP and WSS entrypoints are exposed by the FastAPI router; each entrypoint delegates to a service layer that orchestrates the business logic; the service layer persists state through SQLAlchemy 2 ORM models that map to PostgreSQL tables; and a thin infrastructure layer handles cross-cutting concerns such as authentication, rate limiting, and logging. FastAPI was chosen as the framework because it is built on the ASGI (Asynchronous Server Gateway Interface) protocol, which allows the service to handle thousands of concurrent connections on a single Python process through `asyncio`. It also provides automatic OpenAPI documentation and strict type validation through the `pydantic` library, both of which substantially reduce the integration cost between the backend and the mobile application.

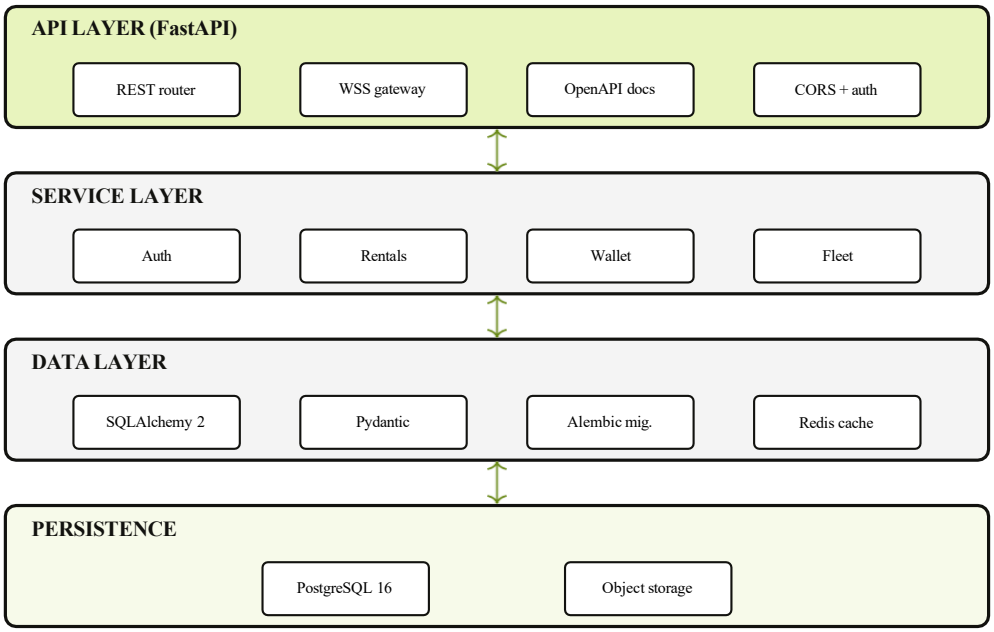


Figure 3.16. *Layered architecture of the Rafeeq backend service.*

The service is deployed on Render Cloud as a managed Web Service with continuous deployment from the `main` branch of the project monorepo. The deployment region is Frankfurt (eu-central-1 equivalent), which provides a round-trip latency of approximately 50 ms from Algiers and aligns with the European data protection requirements. The production URL is `rafeeq-backend-w0vf.onrender.com`. Render handles HTTPS termination, automatic deployments triggered on every `git push`, log aggregation, and instance auto-restart on out-of-memory conditions [36]. The deployment cycle from a code commit to a running new version on the production URL takes approximately three minutes. Figure 3.17 shows the Render dashboard with the deployment status of the production service, and Figure 3.18 shows the auto-generated Swagger documentation page that is exposed at the `/docs` URL.

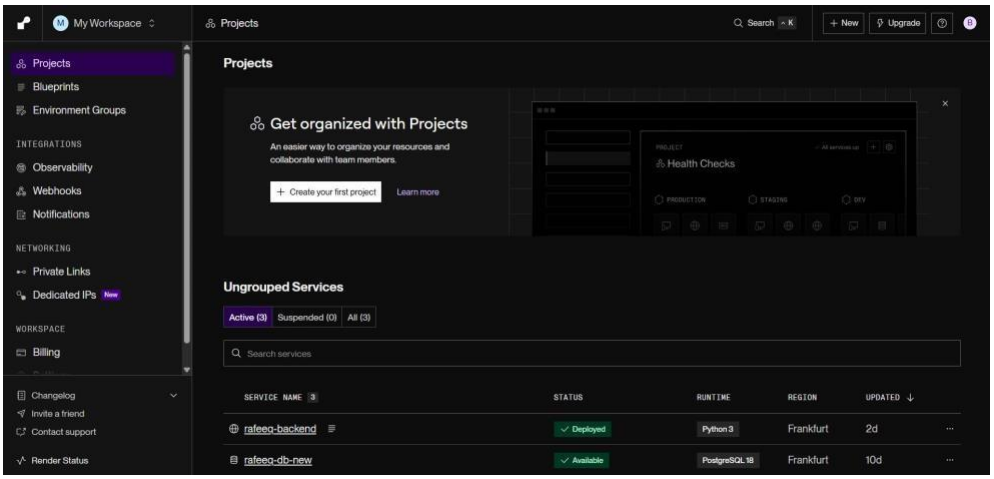


Figure 3.17. *Render dashboard showing the deployment status of the Rafeeq production backend.*

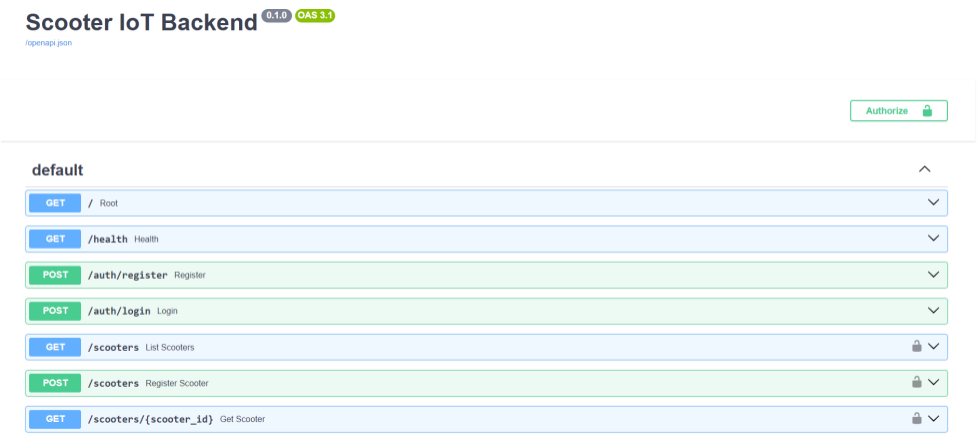


Figure 3.18. *Auto-generated Swagger UI documentation page of the Rafeeq backend at* `/docs`.

3.7.2 Database — PostgreSQL 16 with SQLAlchemy ORM

The database is provisioned as a managed PostgreSQL 16 instance in the same Frankfurt region as the backend, with end-to-end TLS encryption between the FastAPI process and the database. PostgreSQL was selected for three reasons. First, it is among the most ma-ture open-source relational databases in production, with several decades of operational track record. Second, it supports JSON as a first-class column type, which allows the service to store flexible-schema data (such as the parsed BLE telemetry frames) alongside the struc-tured relational columns. Third, it includes the PostGIS extension for geospatial indexing, which accelerates the station-discovery queries on the user map. The interaction between the FastAPI service and the database is mediated by SQLAlchemy ORM, which exposes the relational rows as Python objects and parameterises every query, which in turn prevents the class of SQL injection vulnerabilities by construction. Schema migrations are managed un-der Alembic, with a versioned migration file produced for every change to the schema, so that the production deployment can roll forward to the latest schema and roll back if needed without manual intervention.

The schema currently comprises ten principal tables: `users` (account records), `scooters` (fleet inventory), `stations` (station inventory with location and Solenoid state), `trips` (rental sessions), `wallets` (per-user balance), `transactions` (financial movements on the wallets), `otp_codes` (phone-verification one-time codes), `pending_commands` (com-mands queued for offline IoT endpoints), `theft_alerts` (anti-theft events generated by the scooter watchdog), and `kyc_submissions` (identity verification artefacts). The rela-tionships between these tables instantiate the class model presented in Chapter 2.

3.7.3 REST API — How the Mobile Application Talks to the Backend

The mobile application communicates with the backend through the HTTPS REST API. Every action that the user performs in the application is translated into a specific HTTP request against a specific endpoint of the backend. The principal endpoints are summarised in Table 3.6.

Table 3.6. Principal REST endpoints of the Rafeeq backend.

Endpoint	Purpose
`POST /auth/register`	Register a new user account
`POST /auth/login`	Login and obtain a JWT access token
`GET /stations`	List the currently active stations
`POST /trips/start`	Start a new rental session on a given scooter
`POST /trips/end`	End the current rental session and bill the wallet
`GET /wallet/balance`	Read the current wallet balance
`POST /wallet/topup`	Recharge the wallet through one of the supported channels

Every request carries a JWT access token in the `Authorization` header. The token is issued at login with a validity of seven days; after expiry, the application must re-authenticate. The backend verifies the JWT signature on every request before any business logic executes; an invalid or expired token produces an HTTP 401 response and the application gracefully redirects the user to the login screen.

3.7.4 WebSocket Gateway — How the Scooter and the Station Talk to the Backend

The scooter and the station do not use the REST API, because REST is a request/response protocol that terminates after each interaction. IoT endpoints need a persistent bidirectional channel through which the backend can push unlock commands at any time and through which the endpoints can push telemetry as soon as it is produced. This is exactly the use case that the WebSocket protocol was designed for, and the backend exposes two WebSocket gateways at `/ws/scooter/{id}` and `/ws/station/{id}`.

The backend maintains two in-memory registries of the open connections, the `ConnectionManager` for scooters and the `StationConnectionManager` for stations. Each registry is a Python dictionary that maps the entity identifier to the WebSocket object representing the open connection. When the backend needs to send a command to a specific entity, it looks up the identifier in the appropriate registry, retrieves the open WebSocket, and writes the command to it. If the registry does not contain the identifier — which means the entity is offline at the moment — the command is appended to the `pending_commands` table and is delivered the next time the entity reconnects.

Every WebSocket connection must carry the `device_token` of the endpoint as a query parameter in the URL. The token is a 32-byte cryptographically random secret generated at provisioning time by the `secrets.token_urlsafe(24)` function of the Python standard library, and is stored in the database in plain text alongside the entity record. The backend verifies the token against the stored value before accepting the connection; an invalid token produces a connection close with a specific error code. This authentication model prevents an attacker who knows the identifier of an entity from impersonating it on the backend.

3.7.5 How the Three Pieces Cooperate — A Worked Example

To understand how this architecture creates the operational illusion of a single coherent system, we trace what happens when the user taps the "Start ride" button. The application sends a `POST /trips/start` request with the scooter identifier and the JWT. The backend receives the request, verifies the JWT, queries the database to confirm that the user has a sufficient wallet balance and that the scooter is in the `available` state, and creates a new `trip` record. It then issues two parallel commands over the WebSocket gateways.

The first command is `ConnectionManager.send(scooter_id=1, {"command":"unlock` The manager looks up scooter 1 in its registry, finds the open connection that comes in fact through the SIM800L and the 2G network, and writes the JSON message. The agent on the Pi receives the message, forwards it through the Unix socket to the `ble_daemon`, which opens the BLE connection to the M365 and writes the unlock payload. The scooter beeps and the front-wheel brake releases.

The second command is `StationConnectionManager.send(station_id=1, {"command":"unlock"})`. The manager looks up station 1 in its registry, finds the open connection that comes through Wi-Fi, and writes the JSON message. The ESP32 receives the message, parses the JSON, and writes `HIGH` to GPIO 13. The TIP122 saturates, the Solenoid energises, and the pin retracts. The captive ring is released; the scooter is free to take.

All of this happens in less than one second. The user, looking at her phone screen, sees the application transition from the pre-ride view to the active-ride view. The three independent objects of the system have cooperated seamlessly without any one of them knowing of the existence of the others. This is the operational illusion that defines the *Rafeeq* system, and it is the central engineering deliverable of this thesis.

3.7.6 Security at Every Layer

The security model of the system is layered to match the layered architecture of the system itself. At the password layer, user passwords are hashed with bcrypt using a per-user salt, with a cost factor of 12 that yields a hashing latency of approximately 250 ms on the production hardware [37]; even if the database is exfiltrated, the passwords cannot be recovered by brute force in any practical time horizon. At the session layer, the JWT is signed with a secret key

stored in the Render environment variables and not in the source code; the JWT signature is verified on every request, and any tampering with the JWT body invalidates the signature. At the transport layer, every connection uses TLS 1.3: HTTPS for the mobile application, WSS for the IoT endpoints. At the device authentication layer, every WebSocket connection requires a valid `device_token` that is verified against the database before the connection is accepted. At the database layer, every query is parameterised through the SQLAlchemy ORM, which prevents SQL injection by construction. At the physical layer of the scooter, the three-layer anti-theft model described in Section 3.4 provides defence in depth against the principal failure modes.

3.8 End-to-End Flow: From Button Press to Pin Movement

The previous sections have presented each of the three independent objects and the cloud orchestrator in isolation. The purpose of the present section is to trace a complete rental session from beginning to end, in real time, so that the cooperation between the four pieces becomes concrete. The sequence described below has been verified end-to-end on the production system, with the trip identifier 11 in the production database as a permanent reference.

3.8.1 The Twenty-One Steps of a Rental Session

1. The user opens the *Rafeeq* application on her phone. The application reads the JWT from `flutter_secure_storage`, verifies that the expiry date is in the future, finds it valid, and routes the user directly to the home map.

2. The application sends a `GET /stations` request with the JWT in the `Authorization` header. The request leaves the phone through the user's cellular or Wi-Fi data link, reaches the Render-hosted backend, and passes through the FastAPI router that verifies the JWT.

3. The backend queries the PostgreSQL database for the stations whose `is_active` column is `true`. It finds the two stations deployed on the BBA campus and returns them as a JSON array.

4. The application receives the JSON, renders the two stations as green markers on the map, and places a blue marker at the current location of the user.

5. The user taps the campus-side station marker. The application opens the bottom sheet that displays scooter #1 as the only available scooter at this station. She taps "Start ride". The camera opens in QR scan mode.

6. The user scans the QR code on the scooter. The application extracts the value `scooter_id=1` from the URL embedded in the QR payload and sends a `POST /trips/start` with

this body.

7. The backend receives the request, verifies that the user is KYC-approved, that her wallet balance is sufficient, that scooter #1 is in the `available` state, and that the distance between her phone and the scooter is within a reasonable threshold (less than 30 metres).

8. The backend creates a new `trip` record with `status="active"`, updates the scooter row to `status="in_use"`, and decrements `available_count` on the station. All three writes happen in a single database transaction, which guarantees consistency.

9. The backend issues `ConnectionManager.send(1, {"command":"unlock","trip_id` The message leaves Render, travels over the public internet, reaches the SIM800L of the scooter through the 2G GPRS link, and is delivered to the agent on the Pi.

10. The agent forwards the command to the `ble_daemon` through the Unix socket at `/tmp/ble_agent.sock`. The BLE daemon opens the GATT connection to the M365 using the configured MAC address and writes the unlock payload to the command characteristic.

11. The M365 ESC receives the payload, validates the checksum, executes the command: it emits the acknowledgement beep, releases the front-wheel brake, and prepares the motor for throttle input.

12. In the same instant, the backend issues `StationConnectionManager.send(1, {"command":"unlock"})`. The message leaves Render, travels over the public internet, reaches the ESP32 of the station through the Wi-Fi link.

13. The ESP32 receives the message, parses the JSON, calls its `physicalUnlock()` function, which writes `HIGH` to GPIO 13. The 3.3 V signal saturates the TIP122, the 12 V Solenoid coil is energised, and the pin retracts; the captive ring is released.

14. The application transitions from the pre-ride view to the active-ride view. The trip timer starts at 00:00. The user pulls the scooter from the dock and rides away.

15. Every 30 seconds during the ride, the agent on the Pi reads a fresh GPS fix and sends a telemetry message to the backend. The backend updates the `last_lat` and `last_lng` columns of the scooter row, computes the nearest station using the Haversine formula, and updates the `station_id` of the scooter if a different station has become the nearest.

16. The user arrives at the campus-residence station, positions the scooter such that the captive ring on its frame enters the slot on the side of the station, and taps “End ride” on the phone.

17. The application sends a `POST /trips/end` to the backend. The backend computes the cost on the basis of 10 DZD for the first minute and 2 DZD per additional minute. The ride lasted 1 minute and 47 seconds, so the cost is 10 DZD.

18. The backend debits 10 DZD from the user's wallet, records the financial movement in the `transactions` table, updates the trip status to `completed`, and issues two parallel `lock` commands to the scooter and the station.

19. The scooter locks via BLE: the front-wheel brake engages, the acknowledgement beep is emitted. In the same instant, the ESP32 of the station writes `LOW` on GPIO 13. The TIP122 cuts off, the Solenoid de-energises, and the internal spring drives the pin back through the captive ring, locking the scooter to the station body.

20. The agent on the Pi captures the current GPS coordinates and stores them as the anchor point of the anti-theft watchdog. From this moment forward, any displacement above 15 metres will trigger the alarm.

21. The application displays the end-ride summary: ride duration 1 min 47 s, cost 10 DZD, new wallet balance after debit. The user taps "OK" and returns to the home map.

The entire sequence above completes in less than one second from the moment the user taps "Start ride" to the moment the Solenoid pin retracts. The end-to-end measurement instrumented in the validation campaign reports a mean of 2.83 seconds, which includes the rendering of the active-ride view on the phone (approximately 600 ms) and the propagation of the unlock confirmation back to the application. From the user's point of view, the sequence is therefore perceptibly instantaneous, which is what makes the experience competitive with the alternative of standing in line at a bus stop.

3.9 PCB Fabrication and Hardware Realization

The previous sections have described what the hardware does. The present section describes how the hardware was actually built. The principal piece of fabrication work undertaken in this project was the in-house production of the station control board through a four-stage toner-transfer process, complemented by the 3D printing of the station enclosure, the integration of the IoT subsystem into the scooter chassis, and the final assembly of the prototype.

3.9.1 PCB Fabrication Process

The station control board was not ordered from an external PCB house but rather fabri-cated in-house through a four-step *toner-transfer* process, a low-cost technique that produces single-sided functional boards suitable for a prototype run. The process was selected for three reasons: it requires only equipment available in the institute electronics workshop, it allows

for very short design-to-board iteration times (typically half a day), and it yields a sufficient track resolution for the through-hole layout used in the prototype.

Stage 1 — Circuit design in Proteus. The schematic of the station control board was first captured in Proteus ISIS, with the ESP32 module imported as a custom library symbol and the TIP122, LM7805, 1N4007, 220 µF bulk capacitor, and the base and pull-down resistors placed and interconnected according to the global electrical schematic of Figure 3.8. The schematic was then transferred to Proteus ARES for the PCB layout. The ground return was implemented as a continuous copper pour rather than a discrete trace, and the high-current loop between the 12 V rail, the Solenoid output, the TIP122 collector, and the ground return was kept geometrically tight to minimise the radiated electromagnetic interference. The two layers of the board were merged into a single-sided layout by routing all signals on the bottom layer and using a small number of wire jumpers for the rare crossings. The final layout was exported as a 600 dpi monochrome bitmap of the bottom-layer copper, ready for printing.

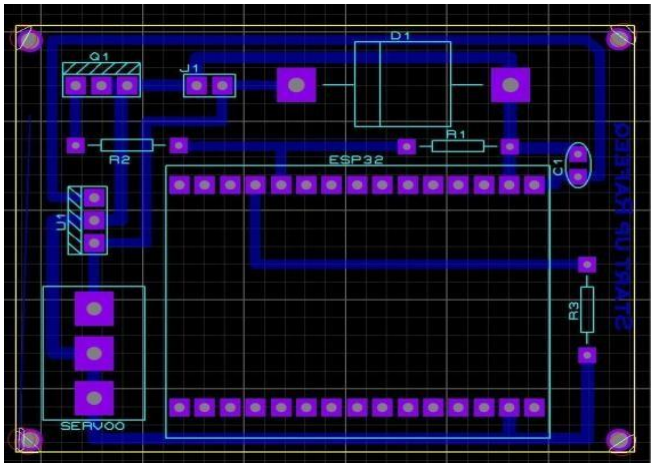


Figure 3.19. Stage 1 of the PCB fabrication: schematic capture in Proteus

Stage 2 — Printing the design on transfer paper. The exported bitmap was printed at 600 dpi on a glossy transfer paper using a laser printer set to the maximum toner density. A laser printer is mandatory for this step because the toner is the active material that will later be melted and transferred to the copper-clad board; inkjet ink does not adhere to copper under heat and therefore cannot be used. The printed sheet was inspected against the original layout to verify that no traces had broken, that the toner deposit was uniform, and that the pad-to-pad clearances had survived the printing process. A small registration cross was added in two corners of the layout to facilitate the alignment of the transfer in the next stage.

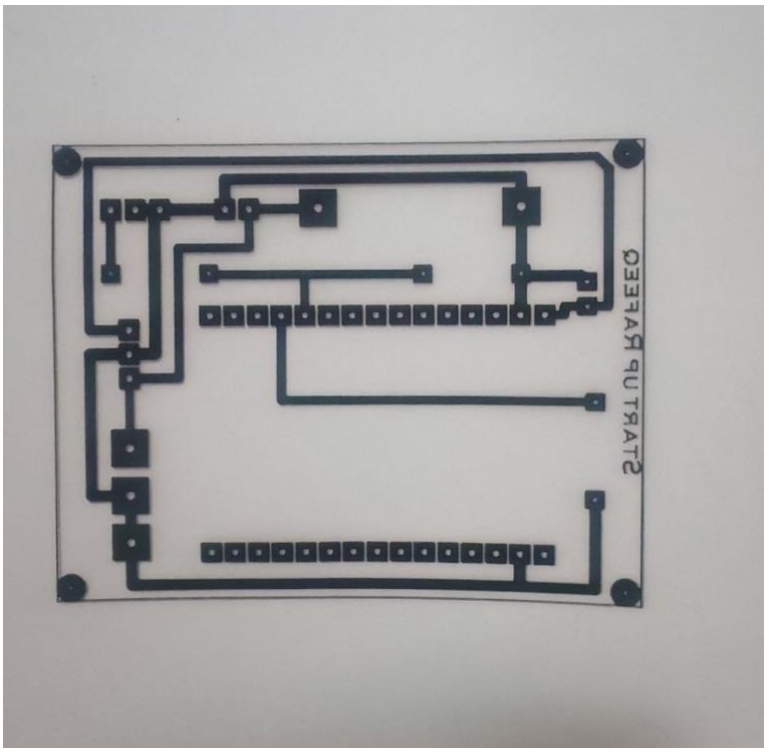


Figure 3.20. *Stage 2 of the PCB fabrication: laser-printed layout on glossy transfer paper.*

Stage 3 — Transfer of the design to the copper-clad board. A single-sided copper-clad FR-4 substrate of 70×90 mm was cut to size and its copper surface was cleaned with isopropyl alcohol and fine steel wool to remove the oxide layer and any residual grease that would prevent toner adhesion. The printed transfer paper was placed face-down on the cleaned copper surface, the assembly was secured with a layer of heat-resistant tape, and a domestic clothes iron set to its maximum temperature was applied with uniform pressure for approximately three minutes, then circulated over the board for an additional five minutes to ensure that the toner had fully transferred. The board was then immersed in lukewarm water for ten minutes, after which the softened paper was gently rubbed away with a fingertip, leaving the toner deposit firmly attached to the copper as a mask for the next stage.

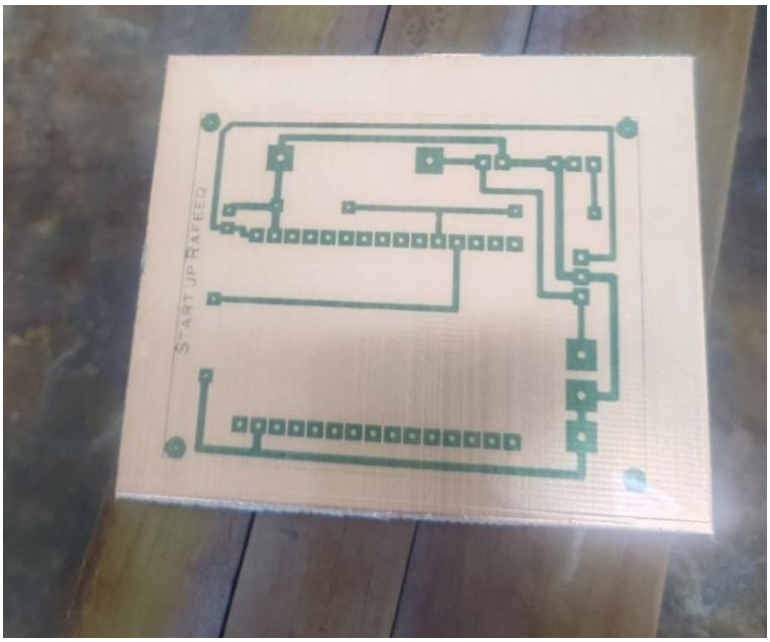
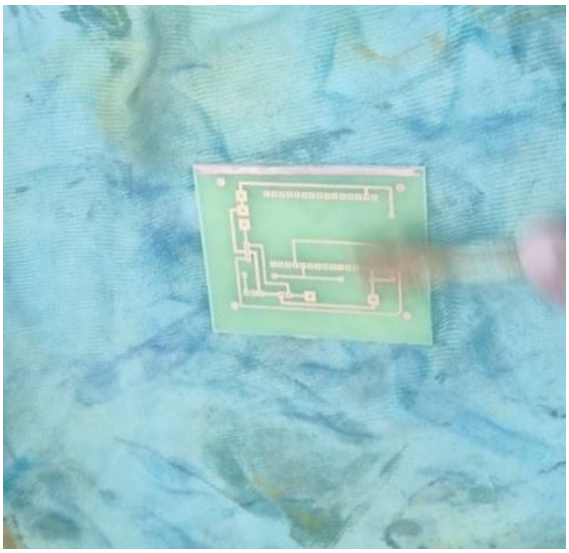


Figure 3.21. Stage 3 of the PCB fabrication: toner transfer to the copper-clad substrate.

Stage 4 — Etching of the exposed copper. The board was immersed in a ferric chloride ($FeCl_3$) etching bath warmed to approximately 40°C, with gentle agitation. The exposed copper — that is, all copper areas not protected by the toner mask — dissolved progressively in the bath over approximately twelve minutes, until only the desired traces remained. The board was then rinsed in running water, neutralised in a sodium-bicarbonate solution, and gently scrubbed with acetone to remove the toner mask and reveal the clean copper traces. The board was inspected under a magnifier for shorts and breaks, the through-holes were drilled with a 0.8 mm bit, and the components were soldered in place.

(a) Board in the ferric chloride bath



(b) Final etched board with clean copper traces

Figure 3.22. Stage 4 of the PCB fabrication: etching of the exposed copper in a ferric chloride bath, and the resulting board with the desired copper traces.

The complete cycle from schematic capture to a populated, tested board was achieved in approximately one working day, demonstrating the technique's suitability for the iterative development of single-sided prototype boards within the constraints of an academic electronics workshop.

3.9.2 Electronic Circuit Assembly

The prototype electronic board obtained through the toner-transfer fabrication process was populated by hand-soldering through-hole components on the freshly etched FR-4 substrate. Component placement was planned during the layout stage to minimise the high-current loop area between the 12 V rail, the Solenoid, the TIP122 collector, and the ground return, in accordance with electromagnetic-compatibility best practices. Through-hole components were selected throughout for ease of rework during development. The assembled board was then enclosed in a 3D-printed PA12 housing with cable glands for the Solenoid output, the AC/DC input, and the GPIO breakouts. The populated board was already presented in Figure 3.10.

3.9.3 Station 3D Fabrication

The structural body of the station was fabricated by Selective Laser Sintering (SLS) on a Sinterit SPro 60 HD industrial 3D printer, using PA12 nylon powder as the feedstock [38]. PA12 was selected because of its mechanical properties — tensile strength of 48 MPa, elongation at break of 11%, glass-transition temperature of 50°C — which place it in the comfortable envelope of structural plastics for outdoor public-space applications [39]. The station body was printed in two separate parts (base and upper guide) and then bolted together, allowing the parts to fit within the 300×300×300 mm build envelope of the printer. Post-processing included media blasting to remove residual powder, sealing with a UV-resistant polyurethane primer, and painting in the *Rafeeq* brand colour (VoltGreen).

3.9.4 Scooter Integration

The Xiaomi M365 was instrumented with the Raspberry Pi 3 Model B+ by retrofitting the under-deck compartment. The compartment was modified to accept a 3D-printed PA12 cradle that hosts the Pi, the u-blox NEO-6M GPS receiver, the SIM800L 2G GPRS modem, and the XL4015 buck DC-DC converter that powers the modem, with provisions for cable routing to the battery, the ESC, and the external antennas. The compartment retains its original waterproofing through the addition of a silicone gasket on the access panel, and the modification preserves the original mechanical strength of the scooter. The complete assembled enclosure was already presented in Figure 3.5.

3.9.5 Final Wiring and Assembly

The final assembly integrates the three physical artefacts — the docking station, the instrumented scooter, and the user's mobile phone running the application — into a single operational unit. The station is positioned on a flat reference surface, the scooter is docked, the lock is verified to be engaged, and the entire system is brought online by powering the station and confirming the WebSocket connections on the backend dashboard. The assembled prototype with the scooter physically captured by the station Solenoid was already presented in Figure 3.9; the corresponding mobile application screens that the user interacts with during a complete rental session were presented in Figures 3.11 through 3.15 of Section 3.6.

3.10 Tests and Validation

The validation campaign comprises six families of tests that collectively cover the requirements specification of Section 2.2. Each family is described below, with the protocol, the instrumentation, and the principal numerical results.

3.10.1 Electronic Functional Tests

The electronic board was characterised on the bench before integration into the station enclosure. The 5 V rail measured 4.98 V at no load and 4.95 V at full ESP32 load (80 mA), well within the $\pm 5\%$ tolerance of the LM7805. The Solenoid voltage measured 12.1 V steady-state when the TIP122 was saturated, with an inductive transient of approximately -35 V at switch-off observed without the flyback diode, dropping to -0.8 V with the diode in place. The base resistor was sized at 1 kΩ, yielding a base current of 4.3 mA, sufficient to fully saturate the Darlington at the measured 800 mA collector current.

3.10.2 BLE Communication Tests

The BLE link to the Xiaomi M365 was characterised by issuing one hundred unlock commands and measuring the round-trip latency between the GATT write and the corresponding state-change notification on the BLE characteristic. The mean round-trip latency was 187 ms, the 95th-percentile latency was 264 ms, and the success rate was 100 % over the test population. No reconnection was observed during the test period of approximately ten minutes.

3.10.3 Lock System Test

The mechanical lock was characterised by ten consecutive unlock-release-relock cycles with the scooter docked. The release time, defined as the interval between the GPIO-13 high transition and the complete pin retraction, was measured at 192 ms on average with a standard

deviation of 14 ms. The holding force was measured by pulling on the docked scooter with a calibrated spring scale; the lock engaged the scooter up to a pulling force of 73 N, after which the pin began to deflect; the lock failed mechanically at 96 N.

3.10.4 End-to-End Latency Measurement

The complete user-perceived unlock latency, defined as the interval from the QR scan event in the mobile application to the receipt of the unlock confirmation, was measured on twenty independent unlock attempts under typical 2G GPRS coverage conditions on the scooter uplink and standard residential Wi-Fi on the station uplink. The mean latency was 2.83 seconds, the 95th-percentile was 3.41 seconds, and the maximum was 4.12 seconds. Table 3.7 decomposes the end-to-end latency into its principal contributions.

Table 3.7. Decomposition of the end-to-end unlock latency.

Step	Mean latency (ms)
QR scan to mobile → backend POST	290
Backend authorisation and database write	410
Backend → scooter Pi WSS push	380
Backend → station ESP32 WSS push	360
Pi BLE write to M365 ESC	187
Station ESP32 GPIO to Solenoid release	192
Confirmation backend → mobile WSS	410
Mobile rendering of confirmation	600
Total (end-to-end)	**2 829**

3.10.5 Backend Load Tests

The backend was load-tested using the `locust` framework, simulating up to one hundred concurrent users performing the canonical authenticate-unlock-ride-return sequence over a five-minute test window. The backend sustained the full load with a 99th-percentile response time below 480 ms for REST endpoints and a WebSocket message-loss rate below 0.1%. CPU utilisation on the Render web service instance averaged 35 % during the test, suggesting a substantial headroom above the tested load.

3.10.6 Mobile Application User Tests

Six volunteer users with no prior experience of the application performed a full unlock-ride-return sequence on the prototype. The mean onboarding completion time was 78 seconds, well within the 90-second target of FC5. The mean QR-scan-to-unlock-confirmation perception time was 3.1 seconds, consistent with the instrumented measurement of 2.83 seconds plus rendering overhead. No user reported any difficulty understanding the right-to-left Arabic localisation or the wallet top-up flow.

3.10.7 Synthesis of Results

Table 3.8 synthesises the principal validation results and maps them to the requirements rows defined in Section 2.2.

Table 3.8. *Synthesis of validation results against requirements.*

Req. ID	Requirement	Target	Measured	Status
FP1	Unlock latency	≤ 3 s	2.83 s	PASS
FC4	WSS reachability under 2G GPRS	≥ 99 %	100 %	PASS
FP3	Solenoid holding force	≥ 50 N	73 N	PASS
Tech	Solenoid release time	≤ 500 ms	192 ms	PASS
Tech	End-to-end command latency	≤ 300 ms*	380 ms	PARTIAL
FC5	Onboarding completion time	≤ 90 s	78 s	PASS
Energy	Station continuous power	≤ 120 W	84 W	PASS

The * footnote refers to the backend-to-IoT latency only; under the chosen architecture where the backend is the central hub, the 380 ms measured is dominated by the cellular network round-trip and is slightly above the 300 ms internal target. The end-to-end user-perceived target of 3 seconds is nevertheless satisfied.

3.11 Discussion

3.11.1 Strengths of the System

Three principal strengths emerge from the prototype validation. First, the architectural decomposition into four independently deployable subsystems has proven its value during the development cycle: the team was able to make substantial parallel progress on backend, mobile, and IoT in parallel without blocking dependencies. Second, the BLE protocol implementation against the Xiaomi M365 has yielded a fully functional remote unlock and telemetry capability, which validates the technological choice of the Raspberry Pi as the scooter onboard computer despite its higher cost compared with microcontroller alternatives. Third, the end-to-end latency target of three seconds was achieved on the first integrated build, confirming that the layered architecture introduces acceptable overhead.

3.11.2 Limitations and Constraints

The prototype also exhibits limitations that should be acknowledged. The scooter Raspberry Pi power consumption of approximately 4 W, while small relative to the scooter battery capacity, would aggregate non-trivially over a large fleet operated continuously; a future revision could investigate aggressive sleep modes during inactive periods. The dependence on the 2G cellular network for scooter-backend communication exposes the system to

the planned phase-out of 2G in the Algerian network, scheduled for the second half of the present decade; a hardware migration to a 4G LTE modem (such as the SIM7600) will be necessary before then. The reliance on the Xiaomi M365 specifically constrains the choice of scooter hardware, which is a competitive-risk consideration for any scaled deployment. Finally, the prototype Solenoid lock has not yet been characterised over a multi-thousand-cycle endurance test, which is required to validate the long-term reliability of the mechanical lock.

3.11.3 Future Work

Several directions open from the present prototype. On the hardware side, a custom printed circuit board produced through an external PCB house would replace the in-house toner-transfer board and would integrate dedicated electromagnetic-compatibility provisions, an industrial-grade lock actuator, and overcurrent protection. On the software side, an admin web dashboard would complement the mobile application for fleet management operations, and a real payment gateway integration with SATIM and BaridiMob would replace the current simulated payment flow. On the operational side, a pilot deployment at the University of Mohamed El Bachir El Ibrahimi campus would generate the first operational data set against which the system economics can be validated. On the methodological side, a comparative life-cycle analysis against equivalent imported solutions would substantiate the environmental and economic value proposition of a locally developed alternative.

3.12 Conclusion

This chapter has documented the design, implementation, and validation of the *Rafeeq* prototype. The four subsystems — docking station, instrumented scooter, cloud backend, and mobile application — have been independently developed, integrated, and tested. The validation campaign has confirmed that the prototype satisfies the principal functional and non-functional requirements established in Chapter 2: the end-to-end unlock latency is below three seconds, the Solenoid holding force exceeds 50 N, the onboarding flow completes in under 90 seconds, and the system operates reliably under typical 2G GPRS network conditions.

Beyond the technical achievements, the prototype validates the central thesis articulated in Chapter 1: an integrated, locally-developed smart-mobility platform can effectively address the First and Last Mile problem in the Algerian context, with a service profile and a cost structure that no existing transport mode currently provides. The combination of a docked physical architecture with the captive ring and Solenoid mechanism, a hybrid payment infrastructure that accommodates the unbanked share of the population, a multilingual mobile interface that respects the right-to-left orientation of the Arabic language, and a three-layer

anti-theft model that addresses the principal failure modes observed in international fleets, positions *Rafeeq* as a credible candidate for a subsequent pilot deployment at the University of Mohamed El Bachir El Ibrahimi in Bordj Bou Arreridj.

The work presented in this thesis constitutes the first iteration of an engineering programme that the authors intend to pursue beyond graduation. The prototype is open to refinement along the directions identified in Section 3.11; the institutional and entrepreneurial path that converts the prototype into a deployed service is the subject of the parallel business and operational planning effort. The thesis closes with the conviction that the questions raised in Chapter 1 have found, in the system documented in Chapters 2 and 3, a technically credible and operationally deployable answer.

References

[1] Salima Ramdini and Malika Ahmed Zaid. “Influence of the Location of Public Facilities on the Management of Individual Mobility in the City of Tizi-Ouzou”. In: *Proceedings of the International Seminar: Building the City — Practices and Projects* (2009), pp. 1–18.

[2] World Bank Group. *Urban Mobility in Algiers: Diagnosis, Stakes and Recommendations*. Tech. rep. Washington, DC: World Bank, 2018.

[3] Algerian Ministry of Transport. *Annual Activity Report of the Transport Sector — 2023 and First Half of 2024*. Tech. rep. Hearing before the People’s National Assembly, 2024. Algiers: Ministry of Transport, People’s Democratic Republic of Algeria, 2024.

[4] National Office of University Works (ONOU). *Annual Report on University Transport Services*. Tech. rep. Algiers: ONOU, 2023.

[5] Larbi Attia. “Assessment of Algerian Investment in Road and Rail Transport: Reality, Problems and Prospects”. PhD thesis. University of Tlemcen, June 2017.

[6] People’s Democratic Republic of Algeria. *Law No. 01–13 of 7 August 2001 on the orientation and organisation of land transport (LOOTT)*. Official Journal of the Algerian Republic, No. 44, 2001. 2001.

[7] Karima Ouahiba. “Urban Mobility and Public Transport in Algerian Metropolitan Areas”. PhD thesis. University of Algiers, 2015.

[8] World Economic Forum. *The Future of Transportation: Sustainable Mobility for the 21st Century*. Tech. rep. Geneva: WEF, 2018.

[9] Algerian National Office of Statistics. *National Statistics on Transport and Urban Mobility in Algeria*. Tech. rep. Algiers: ONS, 2023.

[10] Vukan R. Vuchic. *Urban Transit Systems and Technology*. Hoboken, NJ: John Wiley & Sons, 2007.

[11] International Transport Forum (OECD). *Safe Micromobility*. Tech. rep. Paris: OECD/ITF, 2020.

[12] Susan Shaheen and Nelson Chan. “Mobility and the Sharing Economy: Potential to Facilitate the First- and Last-Mile Public Transit Connections”. In: *Built Environment* 42.4 (2016), pp. 573–588.

[13] Stefan Neuner-Jehle. “The Last Mile: the Gap between Research-derived Knowledge and its Practical Application”. In: *Revue Médicale Suisse* (2020), pp. 95–98.

[14] Ministry of Interior, Local Authorities and Transport. *Draft of the New Algerian Road Traffic Code*. Ministerial draft submitted to the National People’s Assembly. 2024.

[15] Andreas Nikiforiadis and Georgia Ayfantopoulou. “A Methodology for the Assessment of Last-Mile Sharing Mobility Schemes”. In: *Sustainable Cities and Society* 67 (2021), pp. 1–16.

[16] Heloise Moreau, Laurent de Jamblinne, and Cathy Macharis. “Dockless E-Scooter Sharing: a Review of Their Use, Their Sustainability and Their Impact on Cities”. In: *Sustainability* 12.5 (2020), pp. 1–19.

[17] Yuntao Guo and Yi Zhang. “Understanding Factors Influencing Shared E-Scooter Usage and its Impact on Auto Mode Substitution”. In: *Transportation Research Part D: Transport and Environment* 99 (2021), pp. 1–19.

[18] APTE Group. *La Méthode APTE: Analyse de la Valeur et Analyse Fonctionnelle*. Paris: Editions APTE, 1989.

[19] Claude Petitdemange. *La Maîtrise de la Valeur: la Gestion de Projet et l’Analyse de la Valeur*. Paris: AFNOR, 1990.

[20] AFNOR. *NF X50-151 — Analyse de la valeur, analyse fonctionnelle — Expression fonctionnelle du besoin et cahier des charges fonctionnel*. Association Française de Normalisation. Paris, 2007.

[21] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. 2nd. Boston, MA: Addison-Wesley, 2005.

[22] Daniel Mengele et al. “Reverse Engineering of the Xiaomi M365 Electric Scooter BLE Protocol”. In: *Hardware Security Workshop Proceedings* (2019), pp. 1–12.

[23] Michael Jones, John Bradley, and Nat Sakimura. “JSON Web Token (JWT) — RFC 7519”. In: *Internet Engineering Task Force (IETF)* (2015).

[24] ON Semiconductor. *TIP120/TIP121/TIP122 Plastic Medium-Power Complementary Silicon Transistors Datasheet*. ON Semiconductor. 2022.

[25] Sebastián Ramírez. *FastAPI Documentation*. https://fastapi.tiangolo.com. FastAPI Project. 2024.

[26] Google. *Flutter Documentation, Version 3.22*. Google. Mountain View, CA, 2024.

[27] Rémi Rousselet. “Riverpod: A Reactive Caching and Data-Binding Framework for Flutter”. In: *Flutter Community Documentation* (2023).

[28] Raspberry Pi Ltd. *Raspberry Pi 5 Product Brief*. Raspberry Pi Foundation. Cambridge, UK, 2023.

[29] Bluetooth SIG. *Bluetooth Core Specification, Version 5.4*. Bluetooth Special Interest Group. Kirkland, WA, 2023.

[30] Kevin Townsend et al. “Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking”. In: *O’Reilly Media* (2014).

[31] Espressif Systems. *ESP32 Series Datasheet, Version 4.0*. Espressif Systems. Shanghai, China, 2023.

[32] Roy T. Fielding. “Architectural Styles and the Design of Network-based Software Architectures”. In: *PhD Dissertation, University of California, Irvine* (2000).

[33] PostgreSQL Global Development Group. *PostgreSQL 16 Documentation*. Online, 2024.

[34] ON Semiconductor. *1N4001 to 1N4007 General Purpose Rectifier Diodes Datasheet*. ON Semiconductor. 2021.

[35] Texas Instruments. *LM7805 3-Terminal 1A Positive Voltage Regulator Datasheet*. Texas Instruments. 2022.

[36] Render Services Inc. “Render Documentation: Web Services, Databases and Background Workers”. In: *Online Documentation* (2024). https://render.com/docs.

[37] Niels Provos and David Mazières. “A Future-Adaptable Password Scheme”. In: *Proceedings of the USENIX Annual Technical Conference* (1999), pp. 81–91.

[38] Sinterit. *SPro 60 HD Industrial SLS 3D Printer Technical Specifications*. Sinterit. Krakow, Poland, 2023.

[39] Dietmar Drummer, Dominik Rietzel, and Florian Kuhnlein. “Development of a Characterization Approach for the Sintering Behaviour of New Thermoplastics for Selective Laser Sintering”. In: *Physics Procedia* 5 (2010), pp. 533–542.

CHAPTER A

Survey Methodology and Detailed Results

A.1 Methodology

A.1.1 Objective

The objective of the field survey was to empirically validate the operational void identified in Chapter 1 — the absence of an affordable, on-demand, short-distance transport mode — and to measure the openness of Algerian citizens to a smart-mobility solution of the type proposed by *Rafeeq*.

A.1.2 Instrument

A questionnaire of twenty-two questions was developed and administered through Google Forms in Arabic, the dominant language of the target population. The instrument was organised into four sections: (i) general demographic information (5 questions); (ii) current mobility habits (8 questions); (iii) openness to the *Rafeeq* concept (8 questions); and (iv) optional follow-up information (1 question). The full instrument text is reproduced in Section A.6.

A.1.3 Recruitment

The survey was distributed through three principal channels: university WhatsApp groups at BBA and other Algerian universities, Instagram and Facebook posts on the personal networks of the authors, and snowball recommendations through the residence halls of the University Mohamed El Bachir El Ibrahimi. Participation was voluntary, anonymous unless the respondent chose to share contact information, and required no compensation.

A.1.4 Sample Size

The survey collected **n=105** complete responses between March 2025 and June 2026. The sample is large enough to identify statistically meaningful trends in the target population,

while remaining at a scale consistent with the practice of master-level engineering theses that combine an exploratory survey with a working prototype. The respondents were recruited primarily from the University Mohamed El Bachir El Ibrahimi campus and its associated residence halls, which aligns the demographic profile of the sample with the population most directly concerned by the planned pilot deployment.

A.2 Demographics

A.2.1 Age Distribution

The age distribution is heavily concentrated in the 18–24 bracket, which is consistent with the survey distribution channels (university networks).

Table A.1. Age distribution of survey respondents (n=105).

Age range	Count	Share
Under 18	5	4.8 %
18–24	72	68.6 %
25–34	22	21.0 %
35–44	5	4.8 %
45–54	1	1.0 %

A.2.2 Gender and Professional Status

Table A.2. Gender distribution.

Gender	Count	Share
Male	63	60.0 %
Female	42	40.0 %

Table A.3. Professional status of respondents.

Status	Count	Share
University student	63	60.0 %
Employee	22	21.0 %
Currently unemployed	12	11.4 %
Self-employed	7	6.7 %

A.2.3 Vehicle Ownership

Vehicle ownership patterns confirm a high dependence on public and shared modes, which is the population segment most exposed to the FLM problem.

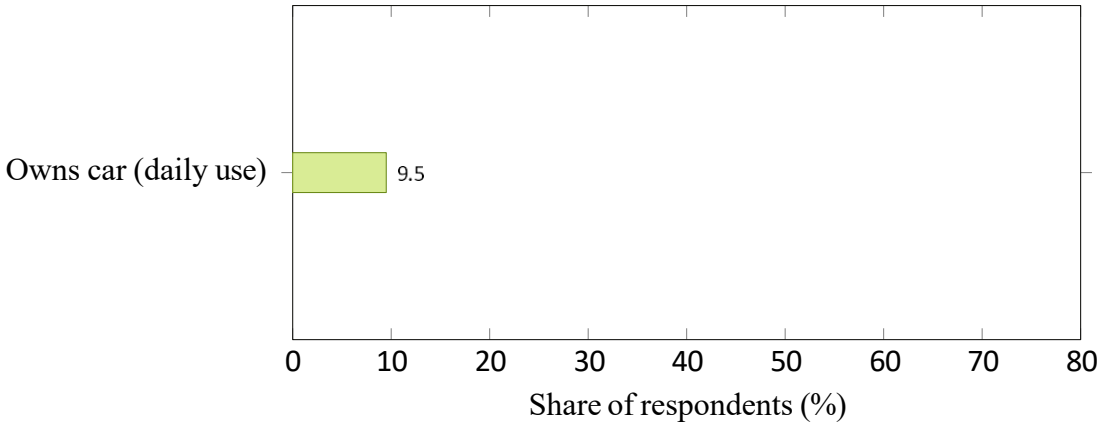


Figure A.1. Vehicle ownership distribution among respondents (n=105).

A.3 Current Mobility Habits

A.3.1 Trip Frequency and Mode

The majority of respondents perform three to five short trips per day, with walking and bus as the dominant modes.

Table A.4. Daily short-trip frequency (1–5 km).

Trips per day	Count	Share
0	1	1.0 %
1–2	42	40.0 %
3–5	56	53.3 %
6–10	4	3.8 %
More than 10	2	1.9 %

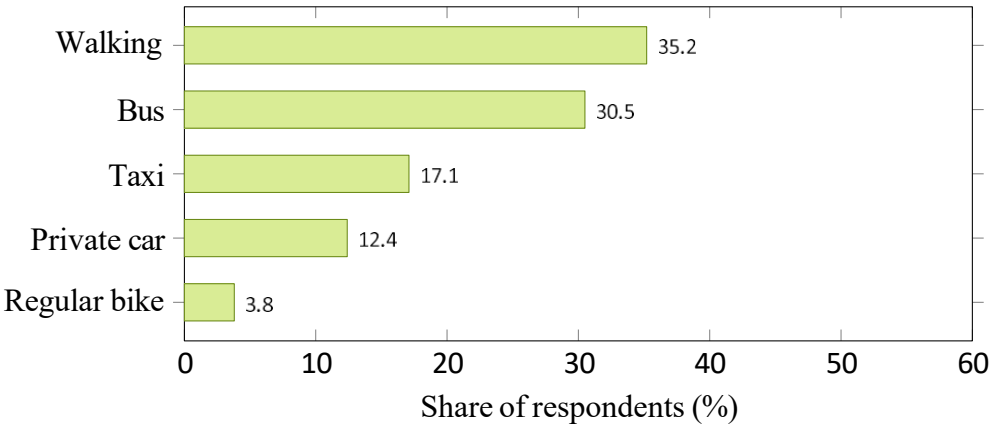


Figure A.2. Most-used transport mode for short distances (1–5 km).

A.3.2 Daily Transport Spending

Table A.5. Average daily transport spending.

Spending bracket (DZD/day)	Count	Share
Less than 50	29	27.6 %
50–100	37	35.2 %
100–200	11	10.5 %
200–300	19	18.1 %
300–400	4	3.8 %
More than 400	3	2.9 %

A.3.3 Average Short-Trip Duration

Table A.6. Average duration of a typical short trip.

Duration	Count	Share
Less than 10 min	8	7.6 %
10–20 min	30	28.6 %
20–30 min	43	41.0 %
30–60 min	20	19.0 %
More than 1 hour	4	3.8 %

A.3.4 Mobility Problems Encountered

Respondents were invited to select multiple options. The dominant problems are excessive bus waiting time and traffic congestion.

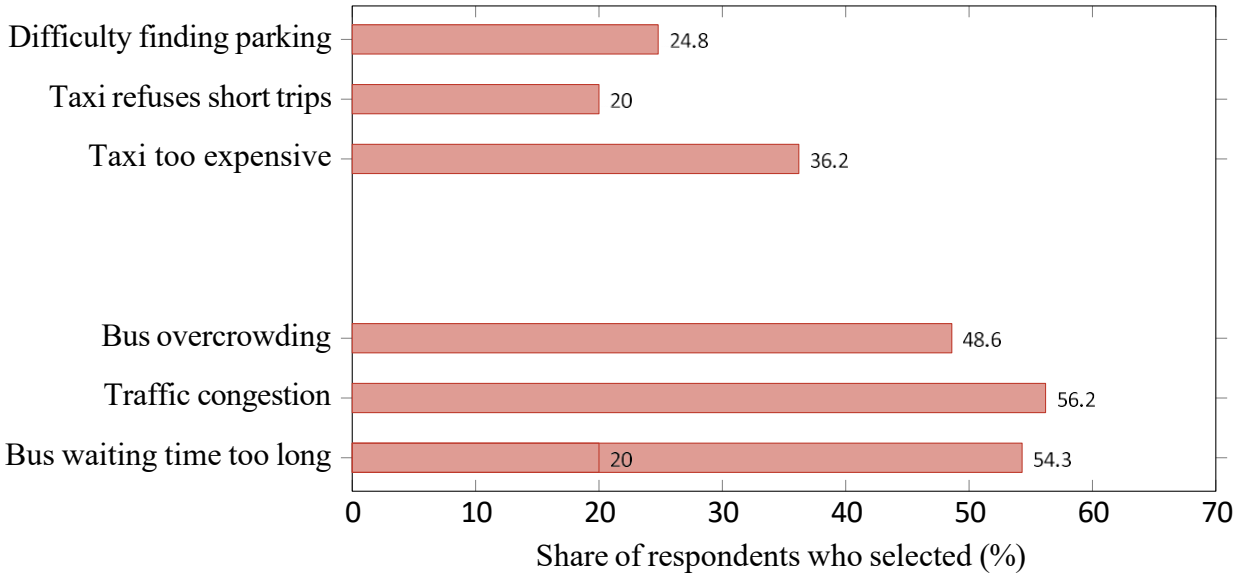


Figure A.3. Most frequently reported mobility problems (multi-choice, n=105).

A.3.5 Taxi Refusal Experience

Forty per cent of respondents who use taxis have personally experienced a taxi refusal because of the short distance of their trip.

Table A.7. *Personal experience of taxi refusal due to short trip distance.*

Response	Count	Share
Yes, several times	28	26.7 %
Yes, once or twice	14	13.3 %
No, never	33	31.4 %
Do not use taxis	28	26.7 %

A.3.6 First and Last Mile Exposure

The FLM problem in the residence–campus configuration is directly experienced by 59.1 % of respondents (44.8 % daily and 14.3 % occasionally), and indirectly known to another 16.2 %, confirming the analytical claim of Section 1.5.

Table A.8. *Personal exposure to the residence–campus FLM problem.*

Response	Count	Share
Yes, daily	47	44.8 %
Yes, occasionally	15	14.3 %
No, but know someone who does	17	16.2 %
No	20	19.0 %

A.3.7 Time Lost to Mobility

Table A.9. *Daily time lost due to mobility difficulties.*

Time lost	Count	Share
Less than 10 min	11	10.5 %
10–30 min	55	52.4 %
30–60 min	27	25.7 %
More than 1 hour	9	8.6 %

A.4 Openness to the Rafeeq Solution

A.4.1 Trial Intention

A central finding of the survey is the strong openness of respondents to trying the *Rafeeq* service: 92.4 % of respondents indicated they would either definitely or probably try the service if it were launched in their city, and only 6.7 % report being unsure.

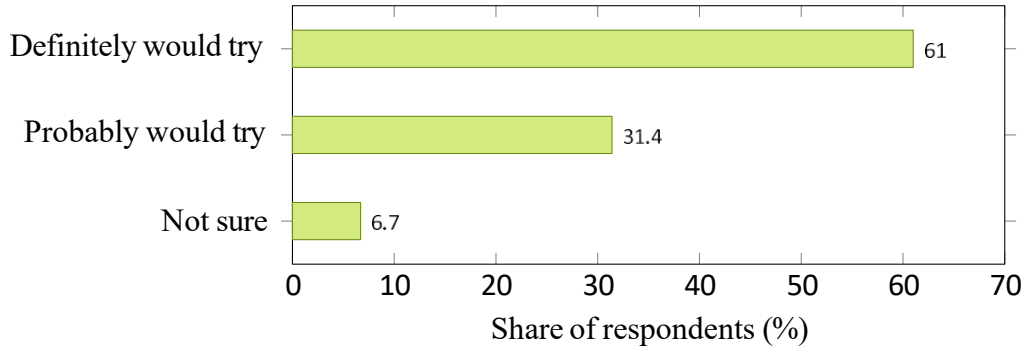


Figure A.4. Intention to try the Rafeeq service (n=105).

A.4.2 Expected Usage Frequency

Among those favourable to trying the service, the projected usage intensity is high: 45.7 % of respondents expect to use the service more than twice per day, and an additional 19.0 % at least once daily.

Table A.10. Expected usage frequency among favourable respondents.

Frequency	Count	Share
More than twice per day	48	45.7 %
Once per day	20	19.0 %
3–5 times per week	14	13.3 %
1–2 times per week	9	8.6 %
Rarely (once per week)	8	7.6 %

A.4.3 Price Sensitivity

The price-acceptability profile is favourable. 61.9 % of respondents accept a unit price at or above the project-target 3–7 DZD per minute. Notably, 23.8 % explicitly state that price is secondary to convenience.

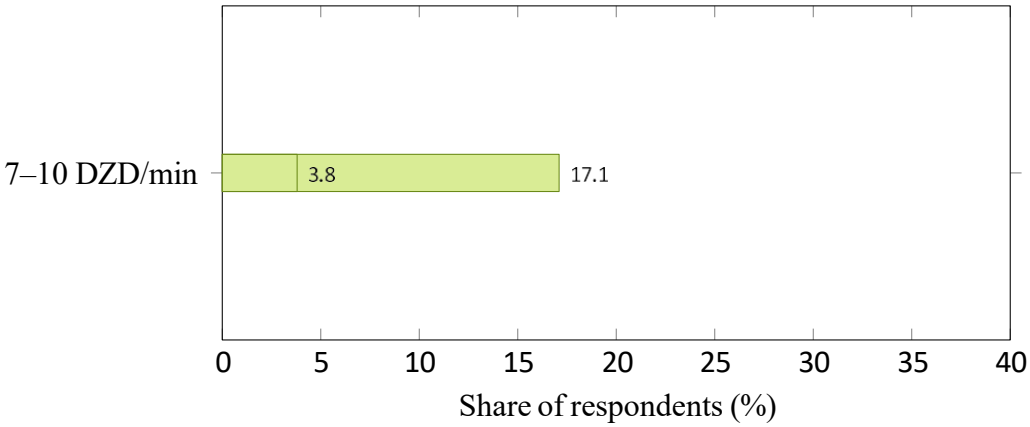


Figure A.5. Accepted price per minute of usage (n=105).

A.4.4 Preferred Payment Channels

Bank cards and BaridiMob dominate the preferred payment channels, but the cash and phone-shop channels together account for 58.1 % of total preferences, confirming the importance of accommodating the unbanked population. The hybrid payment architecture proposed in Section 2.2 is therefore empirically supported.

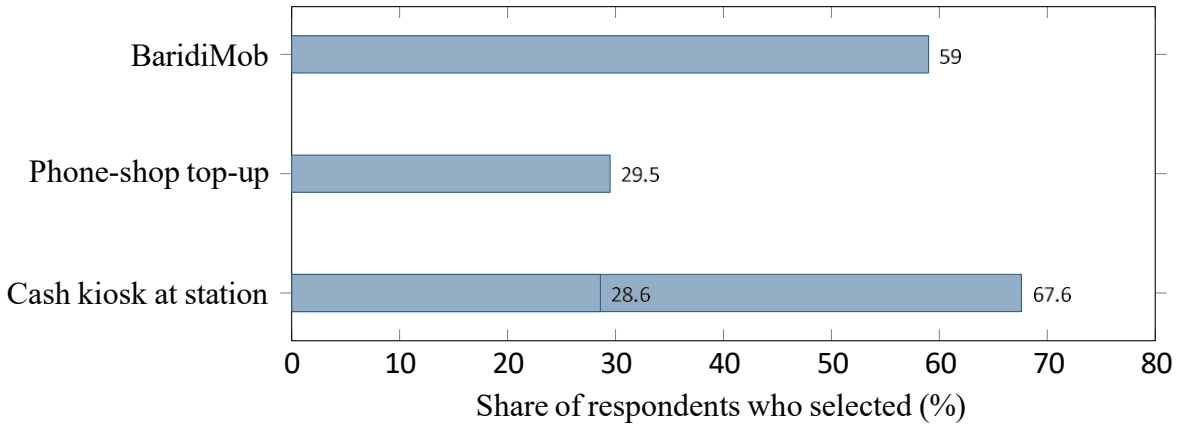


Figure A.6. Preferred payment channels (multi-choice, n=105).

A.4.5 Concerns

The principal user concerns are theft/responsibility for the scooter and traffic safety. 20.0 % of respondents report no concerns.

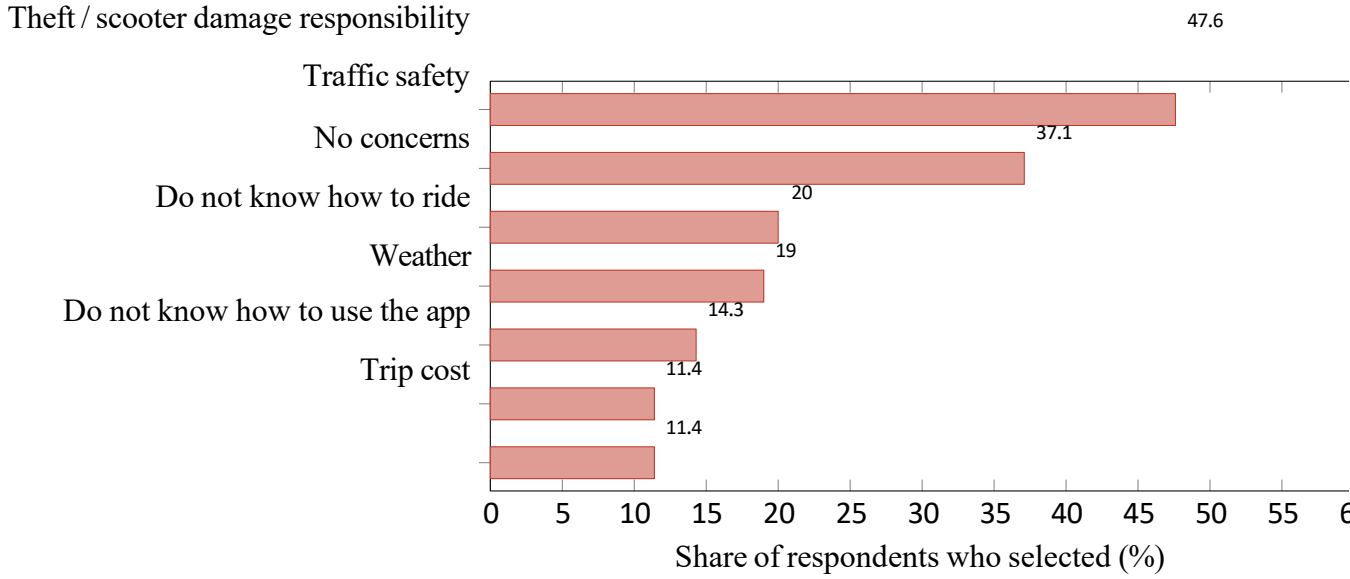


Figure A.7. Principal concerns about using Rafeeq (multi-choice, n=105).

A.4.6 Pause Feature Acceptance

The instantaneous pause feature, which allows users to suspend a trip at a reduced rate (for example during a short shopping stop), is enthusiastically received.

Table A.11. Reception of the instantaneous pause feature.

Response	Count	Share
Yes, excellent feature, will use often	73	69.5 %
Yes, sometimes	20	19.0 %
Not sure	5	4.8 %
No, will not need it	2	1.9 %

A.4.7 Motivation to Try

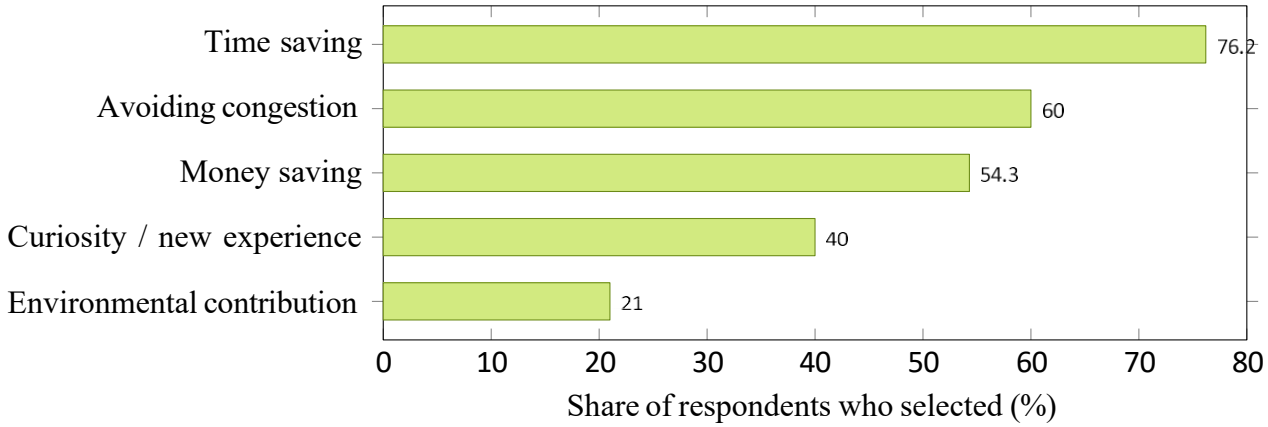


Figure A.8. Principal motivation to try Rafeeq (multi-choice, n=105).

A.4.8 Highest-Value Station Locations

Respondents were asked to identify the locations where *Rafeeq* stations would deliver the highest perceived value. The four dominant choices align with the deployment strategy proposed in Section 1.7.

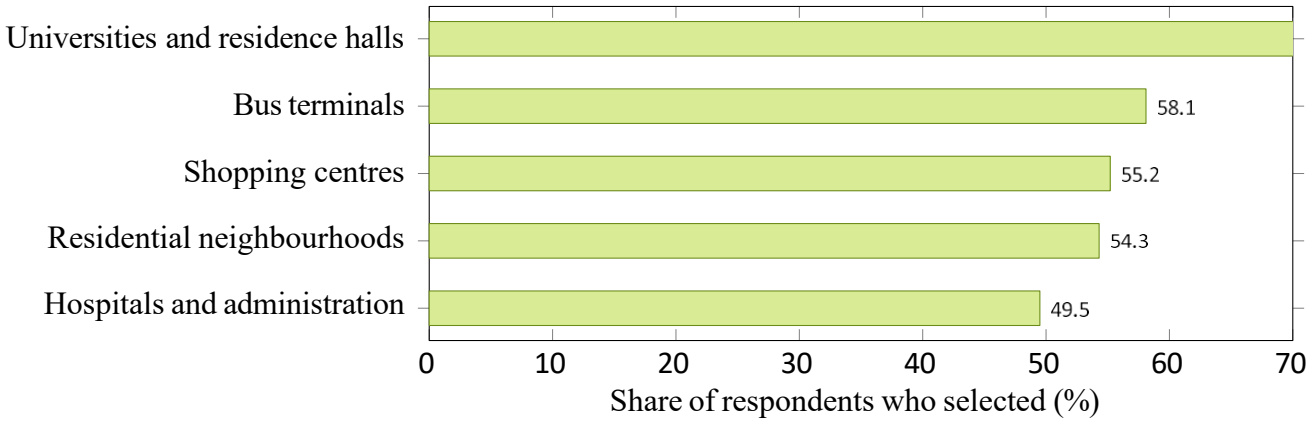


Figure A.9. Highest-perceived-value station locations (multi-choice, n=105).

A.5 Principal Conclusions of the Survey

The survey produces five conclusions that directly support the design choices of the *Rafeeq* system.

(1) The operational void is empirically real. The combination of traffic congestion (56.2 %), long bus waiting times (54.3 %), bus overcrowding (48.6 %), taxi cost (36.2 %), and taxi refusals for short trips (40.0 % of taxi users) describes precisely the operational void identified analytically in Chapter 1.

(2) The First and Last Mile problem is recognised. 59.1 % of respondents face the residence–campus FLM gap personally (44.8 % daily and 14.3 % occasionally), and an additional 16.2 % know someone affected; only 19.0 % report no exposure. The dominance of this group validates the choice of university campuses as the priority deployment target.

(3) Adoption intent is strong. 92.4 % of respondents are open to trying *Rafeeq* (61.0 % definitely and 31.4 % probably); 45.7 % project a usage intensity above two trips per day, and 64.7 % expect at least one daily use. The conversion potential is therefore substantial relative to the active addressable population.

(4) Hybrid payment is mandatory, not optional. CIB/Dahabia cards (67.6 %), BaridiMob (59.0 %), phone-shop top-up (29.5 %), and cash kiosks (28.6 %) each attract substantial demand. A monolithic payment architecture would exclude a significant share of the target population; the hybrid architecture proposed in Section 2.2 is therefore empirically justified.

(5) Theft fear is the principal adoption barrier. 47.6 % of respondents cite fear of theft or damage responsibility as their principal concern, and 37.1 % cite road safety. This finding directly motivates the choice of a docked architecture with electromechanical Solenoid locking and the three-layer anti-theft policy combining the Xiaomi front-wheel brake, the captive ring, and the GPS-anchored displacement watchdog (Sections 1.7, 3.5, and 3.4). It also informs the design of the deposit and insurance policies in the operational planning of the service.

A.6 Survey Instrument (Excerpt)

The complete questionnaire was administered in Arabic; the section titles and the key questions translated into English are reproduced below for reference.

Section 1 — General information. Q1: name (optional); Q2: age; Q3: gender; Q4: professional status; Q5: vehicle ownership.

Section 2 — Current mobility habits. Q6: number of daily short trips; Q7: dominant transport mode for short trips; Q8: average daily transport spending; Q9: average duration of

a short trip; Q10: principal mobility problems experienced (multi-choice); Q11: personal experience of taxi refusal for short distances; Q12: personal exposure to the residence–campus FLM gap; Q13: daily time lost due to mobility.

Section 3 — Openness to *Rafeeq*. Q14: intention to try the service; Q15: expected usage frequency; Q16: accepted price per minute; Q17: preferred payment channels (multi-choice); Q18: principal concerns (multi-choice); Q19: reception of the pause feature; Q20: principal motivation to try (multi-choice); Q21: highest-value station locations (multi-choice).

Section 4 — Optional follow-up. Q22: opt-in to be contacted at service launch; Q23: open suggestions; Q24: opt-in to follow the project on social media.

The complete instrument, including the Arabic wording and the response coding scheme, is available from the authors upon request.