BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Peer-to-peer Car Sharing Implemented in a mobile application

Supervisor
lect. dr. Cojocar Dan

Author
Iacob Victor-Ștefan

2023

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ÎN LIMBA ENGLEZĂ

LUCRARE DE LICENŢĂ

Închirierea mașinilor de la un user la altul
Concept implementat într-o aplicație pentru telefoane mobile

Conducător științific
lect. dr. Cojocar Dan

Absolvent
Iacob Victor-Ștefan

2023

ABSTRACT

Traffic congestion has become a major concern in today's society and figuring out the best way to tackle this issue still remains a big challenge. A solution to combat it is presented in this thesis which focuses on the development of a mobile application centered around the concept of 'Peer-to-peer car sharing'. The application aims to provide a platform where individuals can share their personal vehicles for rental purposes, therefore optimizing existing transportation resources and reducing the number of cars on the road. The mobile application utilizes modern technologies and user-friendly interfaces in order to create a smooth interaction within the 'sharing' community. By exploiting the power of smartphones connectivity and location services, users can easily find the most suitable car for their preferences.

The thesis explores the implementation details, including the choice of frameworks and technologies used in the development process. Moreover, it discusses the benefits and challenges associated with the 'Peer-to-peer car sharing' concept and explores potential future improvements to features and user experience. On the whole, this research contributes to the field of sustainable transportation by promoting a more efficient method of utilizing existing transportation resources and offering a reasonable solution to traffic congestion.

Contents

List of Figures

Chapter 1

Introduction

In recent times, the continuous development of technology has managed to revolutionize various domains and solve problems for which we did not even have an answer long ago. In this thesis, we will discuss the possibility through which modern technology, specifically a mobile application, could revolutionize the way we approach personal transportation.

Right from the title, it can be noticed the concept of 'Peer-to-peer car sharing', which represents the topic around which this thesis is built. This concept is based on the idea of allowing two individuals establish a car rental process without the need of a third party, which is often a rental company. By removing the third instance from the calculation, various benefits can be created, such as cost reduction of the service or the variety of options from which the renter can choose. More about these advantages will be discussed in the following theoretical chapters of this work.

In order to contextualize the idea of 'Peer-to-peer car sharing', the development of a mobile application has been achieved. This application helps users to find a suitable car for their preferences, or allow the listing of their own car for rental purposes. The functionalities of this application, the underlying architecture and the significant advantages that mobile applications possess over traditional web apps are all widely approached into this thesis.

1.1 Addressing the issue

Nowadays, it is quite clear that traffic is a real problem in the big cities of Romania, thing that does not seem to be remedied in the near future. Due to the excessive number of cars present on streets every day, people end up wasting valuable moments of their life waiting for the green light. Along with the time lost in traffic, there is also the issue of environmental pollution and the money wasted on fuel due to high consumption cause by frequent stops and starts.

People in charge have started to apply different methods in order to reduce the number of cars on the road. Unfortunately, public transportation and infrastructure, in general, are still at an unfavorable level in Romania, and people are constantly forced to rely on their cars to travel from point A to point B.

However, the real problem, in fact, consists in the way people purchase and utilize personal cars. Many individuals buy a car with the hope of using it frequently, but in reality, it remains untouched for over 90% of the time. This situation leads to the occupation of parking spaces that could be utilized by other people who genuinely need the car. Additionally, there are monthly or annual costs that increases as the car is getting old, causing owners to pay substantial amounts for a vehicle that is rarely used throughout the year.

The introduction of 'Peer-to-peer car sharing' concept may help in reducing the cars on streets. People could offer their own cars for renting and at the same time, potential renters can enjoy the benefits of using a car without the need to purchase one and bear the associated costs, especially for relatively short distances.

By adopting this approach, it creates an opportunity to utilize existing transportation resources more efficiently, reduce the number of unused cars, and provide an accessible and convenient alternative to purchasing a personal vehicle.

1.2 Motivation

As a student who had to commute from another city for my studies, I mostly relied on my personal car for the journeys between 'home' and the university city. However, since public transportation was free during the years of study, and my personal points of interest in the city were relatively close by, I did not need to use my car frequently. This led me to think abut a way to make my care more useful, both for myself and for others, and that is when I came up with the idea of 'Peer-to-peer car sharing'.

By implementing this concept, not only would my car no longer remain idle, taking up valuable parking space, but it would also transform an active expense into a passive income source, which is a significant advantage for a student with no source of income. This motivated me to explore the potential of 'Peer-to-peer car sharing' and develop it further in this thesis.

1.3 Thesis structure

The structure of this thesis consists of an introduction where the addressed problem is presented and the motivation behind the development is argued. The concept of 'Peer-to-peer car sharing' will be widely discussed, along with an analysis of existing similar application in the field in Chapter 2. Then, the focus will shift to mobile technologies and the benefits they bring to the development of this application in Chapter 3. The application itself, its functionalities, the underlying architecture and other practical details will be presented in Chapter 4, while conclusions regarding this thesis and potential future development directions will be explored in the final chapter.

Chapter 2

Car Sharing and Competitive Landscape

In this chapter, the concept of 'Peer-to-peer car sharing' will be presented in more detail. After a brief introduction, the idea will be examined from an economical and environmental perspective. Potential obstacles that can be encountered during the implementation of this concept are also taken into consideration and discussed into this theoretical chapter. Lastly, there will be an analysis of some applications within the same domain in order to gain a better understanding of the current state of the industry and the objectives that need to be achieved.

2.1 The Importance of Peer-to-Peer Car Sharing

'Peer-to-peer car sharing' has emerged as an innovative concept that has the potential to transform the way personal transportation is approached. This type of service allows individuals to rent out their personal vehicles to others, creating a network of vehicles that can be accessed by those in need of transportation.

The advantages of 'Peer-to-peer car sharing' are numerous and substantial. It is worth noting the most important ones out of them as highlighted in the article published by Portland State University [5]:

- **Fleet efficiency:** By allowing individuals to rent their personal vehicles that tend to remain idle for over 90% of the time, it benefits both the car owner who can earn extra income and the individuals who can easily find and rent a car when needed, without worrying about extra expenses of owning the car. It promotes resource efficiency and helps reducing the number of cars on the streets.

- **Lower cost:** By connecting car owners directly with individuals in need of a vehicle, the costs associated with the third instance, which most of the time is a rental agency, are eliminated. This results into lower rental rates for users, making the service more accessible. Also, it typically offers more flexible rental duration and availability in multiple locations, since there is no boundary in the rental process.

- **Greater access for lower-income households:** By offering greater accessibility at a reasonable cost, it enables members who are price-sensitive to acces employment and other opportunities that may not be easily reachable through public transportation alone.

2.1.1 Economical analysis

'Sharing' has always been a part of human nature. It is completely normal to host a friend in the spare room of your house or giving him a ride if he needs your help. However, the core of 'Sharing economy' is built on the premise of sharing your goods with strangers in order to generate a passive income. As stated in Arun Sundararajan' book [1], the concept of 'Sharing economy' is gaining popularity in the current era, due to the advancements in technology. With applications like Airbnb - where an individual can share his spare room - or BlaBlaCar - where an individual can share a ride with others - people can now use this new concept in their favor. Peer-to-peer car sharing is also a branch of this 'Sharing economy' concept, which allows car owners to unlock the economical potential of their idle vehicles while fulfilling the needs of others. It has the potential to significantly impact traditional car rental markets and generate cost savings for consumers.

Peer-to-peer car sharing services provide a cost-effective alternative to traditional car ownership. In addition to the initial purchase price, car ownership implies ongoing expenses such as insurance, maintenance, and fuel. These costs can be a significant financial burden for many individuals. With peer-to-peer car sharing, individuals can rent a vehicle only when they need one, rather than facing the ongoing costs of car ownership. This may be a more affordable option, especially for those who do not drive frequently. By only renting a car when necessary, individuals can substantially reduce their transportation expenses and free up funds for other expenditures.

Moreover, this type of service typically offer more affordable rental rates than conventional car rental agencies. This is due to the fact that peer-to-peer car sharing companies lack the overhead expenses associated with possessing and maintaining a fleet of vehicles. Instead, vehicle owners are responsible for their own maintenance, which reduces costs for both the owner and the person leasing the vehicle.

Also, P2P car sharing services frequently offer a variety of vehicle categories, such as economy cars, luxury cars, and specialty vehicles. This enables renters to select the most suitable car for their specific requirements, as opposed to being forced to rent a vehicle that may not be the 'best fit'.

Peer-to-peer car sharing can be an effective way for car owners to generate additional income. For example, a small sedan like a Volkswagen Jetta listed on 'Turo' can generate approximately $40 per day in rental income. If the car is rented for 250 out of 365 days in a year, the owner can generate around $8,000 in revenue annually. After factoring in the estimated cost of depreciation for a vehicle driven 15,000km/year, which is around $2,000 including taxes, the owner can still be on profit with over $5,000. This amount is significant, considering that a brand new Volkswagen Jetta costs around $30,000, and it can take up to five or six years to earn the equivalent amount by renting the car. Therefore, P2P car sharing can provide car owners with an attractive source of income while also increasing the efficiency of vehicle use and promoting the sharing economy.

Last but not least, there is a potential to create new job opportunities as there may be an increased demand for individuals to provide services like cleaning and maintaining vehicles in order to keep the renters satisfied. This is a real opportunity for entrepreneurs and small business owners to provide related services such as vehicle detailing, advertising, and marketing. As a result, the growth of peer-to-peer car sharing services may provide a significant contribution to the economy.

2.1.2 Environmental analysis

We are currently living in an era where air pollution and time wasted in traffic have become two of the most significant issues in the day-to-day lives of individuals. The effects of air pollution on public health and the environment have become a growing concern, and cities around the globe are struggling to combat the effects of vehicle emissions. In the meantime, the amount of time wasted in traffic not only contributes to increased levels of stress and frustration, but also has economic consequences, resulting in decreased productivity and higher transportation costs. Peer-to-peer car sharing services offers a significant potential to reduce the environmental impacts of traditional car use by reducing the number of cars on the road and promoting more efficient use of existing vehicles.

According to the article written by Robert Hampshire and Srinath Sinha, most of the private vehicles are inactive for over 90% of the day. [10] By contrast, peer-to-peer car sharing services can increase the utilization rate of existing vehicles, meaning that there may be a corresponding decrease in the demand for new cars overall. This would have a major effect on the environment, as car production accounts for

approximately 6 percent of global CO2 emissions from energy use, making it a significant contributor to climate change.

A recent study by researchers at the University of California, Berkeley, stated that a single shared vehicle can replace anywhere from nine to thirteen privately-owned vehicles. 2.1 This is due to the fact that the most of privately owned vehicles are inactive the majority of the time. By sharing vehicles, individuals can reduce the number of automobiles on the road, thereby reducing traffic congestion, air pollution, and greenhouse gas emissions.

Lastly, the P2P car sharing services also promote the use of electric and hybrid vehicles, which are more eco-friendly than conventional gasoline-powered vehicles. Many platforms allow users to rent a variety of electric and hybrid vehicles, making it simpler for individuals to test-drive these vehicles before purchasing one.

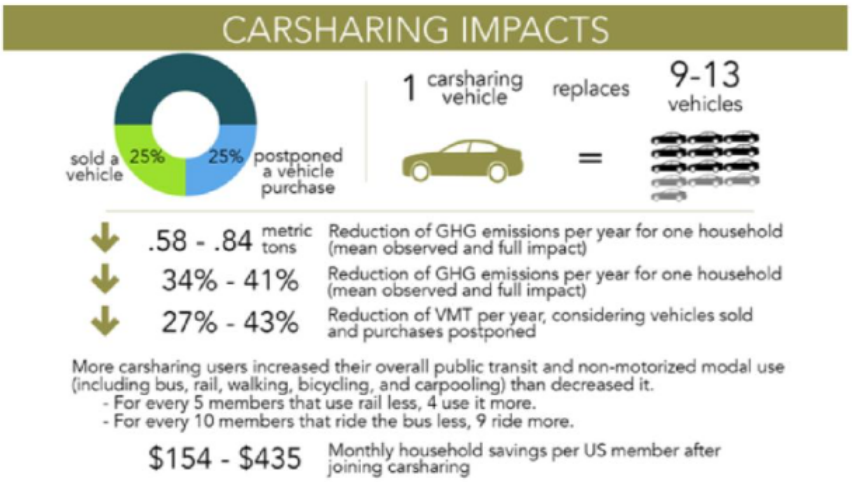

Figure 2.1: Impact of carsharing service [15]

2.1.3 Obstacles

Now that we have discussed the benefits and advantages of peer-to-peer car sharing services, it's essential to consider some of the obstacles that may arise with this innovative approach. By examining these potential obstacles, we hope to provide a more complete understanding of the peer-to-peer car sharing landscape and offer insights into how these obstacles can be overcome to ensure the success of this emerging trend in the sharing economy.

In the year of 2011 researchers from USA asked both personal vehicle sharing and traditional carsharing operators to identify the top three barriers to widespread adoption of personal vehicle sharing. [19] The results of the survey indicated that insurance and a willingness to rent personal vehicles are the most common obstacles to the growth of personal vehicle sharing. 2.2

Insurance has been a significant challenge for peer-to-peer carsharing services around the globe. This is because traditional auto insurance policies did not typically cover commercial use of personal vehicles, leaving car owners exposed to liability risks. Initially, the cost of an insurance that should cover personal vehicle sharing policies was a big inconvenient, since it took away somewhere between 20-25% of operator's overall costs. In 2011, the annual cost of insurance could amount to as much as $2,500, which represented a significant expense. However, as the industry has developed and more data has become available, there was an important change in how the insurance was assessed. By moving on from per-hour to per-km driven insurance which should reflect better the vehicle exposure rate, the cost has dropped to almost 50% and more people were getting into carsharing industry.

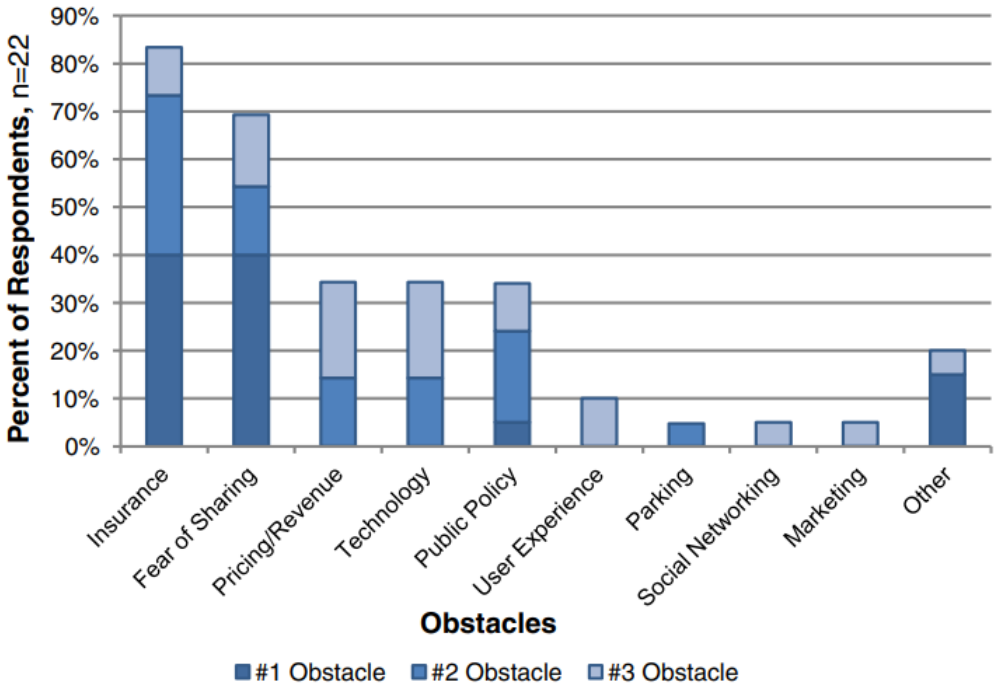

Figure 2.2: Barriers to adoption and expansion of personal vehicle sharing [19]

Moving further, the second obstacle might be the people's reluctance of sharing their personal vehicles with strangers. This is mainly due to safety and security concerns, which can be a challenge for the industry to overcome. In order to address these issues, various car sharing applications have set up a strict verification process prior to allowing users to explore rental options. This process typically requires the user to provide a valid driver's license and a personal identification document in order to verify their identity. By implementing this method, car sharing applications can effectively eliminate potential bots or thieves, thereby establishing a safe environment for both vehicle owners and renters. This verification processes offers users with a sense of security and can help to build trust within the car sharing

community.

An article published in 2020 examined the factors influencing a person's willingness to rent out their personal vehicle. The study's findings 2.3 indicate that older males with no children, who have a stable income and own one or two extra cars are more likely to consider carsharing as an opportunity for them to earn some extra income, while helping others to travel from A to B in a more convenient way. But, times has changed a little, since the COVID-19 pandemic in 2020 and more people start to spend more time working from home instead of going to the actual building of the company. As a result, the likelihood of using personal vehicles has decreased significantly. Therefore, it is reasonable to consider adjusting the following statistics by an additional 10 percent to reflect this change, which leaves us with a small group of individuals who are not inclined towards renting out their personal vehicles.

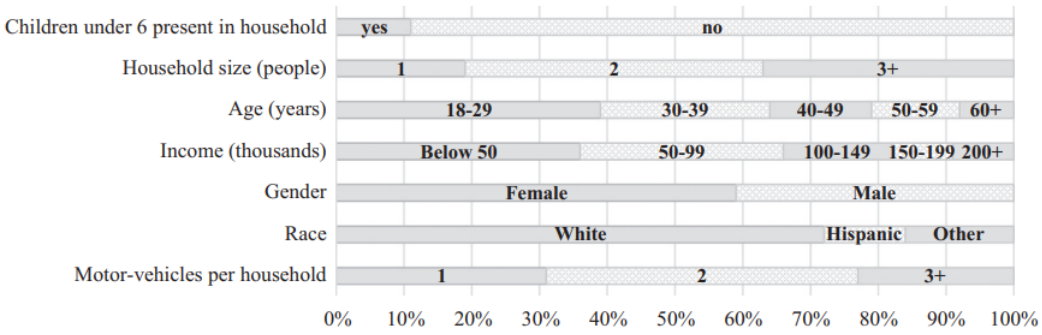

Figure 2.3: Willingness to rent personal vehicles to others [3]

To summarize this last section, I think that the potential obstacles could be overcome if it is for the adoption of some proper measures and by understanding the big picture of the carsharing services which can be a real advantage for both people and the planet Earth.

2.2 Competition analysis

Before jumping into the paragraphs describing the features and technologies that existing car sharing applications are using, it is worth to say that this domain can be divided in two main parts: private fleet and peer-to-peer car sharing services. The first one is the most popular in our country and it is a very useful alternative if an individual does not own a car and needs to go from A to B without worrying that he must get the car back to A. That is because, services often charge per minute basis and the car can be parked in multiple places around the city. However, it is noteworthy that peer-to-peer car sharing services are yet to be introduced in Romania, creating an opportunity for me to implement this idea. By developing my own

car sharing application, I am looking to offer individuals the opportunity to share their privately owned vehicles with those in need, promoting a collaborative and sustainable transportation solution.

In order to be more precise about the analysis, I have gathered some useful information about five of the most successful car sharing applications that are already on the market. Three of them are available in Romania and they are using the private fleet method. The last two are from abroad, but I think that they must be present into the comparison, because of their longevity at the top of the industry. I invite you to take a look at the advantages and disadvantages of each in the following section of my thesis.

The first thing that pops up from the comparison above 2.1 is the fact that every app out of those five requires a client validation. That is because individuals are accessing cars directly through the mobile application most of the times, without interacting with the actual owner, so a high level of security must be established. To achieve this, car sharing platforms typically require users to upload essential documents such as a driver's license and a valid credit/debit card. 2.4 In certain cases, apps may request an additional selfie or passport/personal ID in order to ensure the authenticity of users and minimize the risk of identity theft. By comparing the submitted documents with the user's physical appearance, car sharing platforms can establish a higher level of confidence in the identity of their users, reducing the potential for fraudulent activities and increasing overall security.

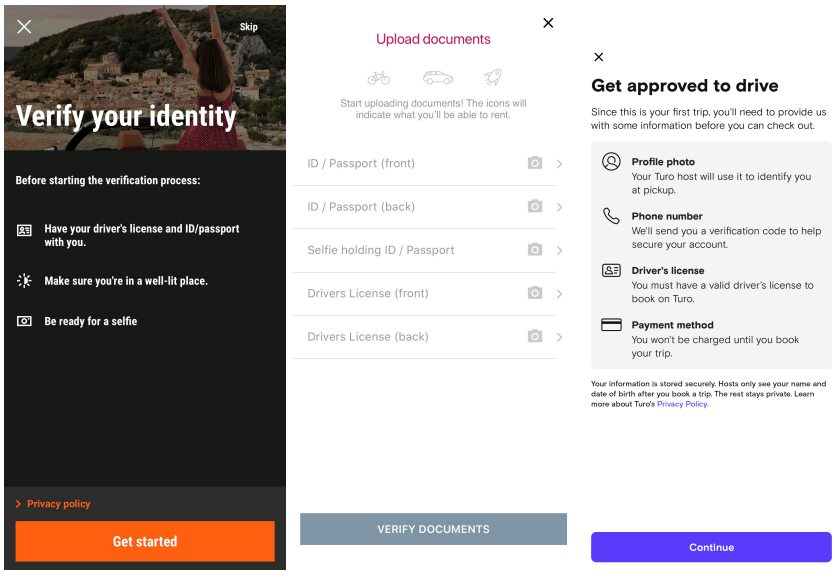

Figure 2.4: Validation process on different applications (Sixt [20]/CityLink [4]/Turo [11])

An important common aspect that plays a vital role is the availability of the applications on both Android and iOS platforms. Whether it is a hybrid or a native approach, this cross-platform accessibility guarantees that users can seamlessly access

Name	Available in Romania	Fleet	Android /iOS	Additional info
CityLink	Yes, only in Brasov and Bucharest	Private, limited selection of cars, including Toyota and Dacia models	Both	it requires a valid passport, a selfie holding the passport and a valid drivers license; 50 ron minimum fare; 1 extra km = 1 ron (when trip distance exceeds daily included 100 kilometers)
Spark	Yes, only in Bucharest	Private, it offers only electric cars	Both	it requires a credit card and a valid drivers license; it charges 200 ron to validate the credit card and it gives back 250 ron as a voucher that can be used for rentals; it has a loyalty bonus points system
eGo	Yes, only in Bucharest	Private, it offers only electric cars	Both	exploring cars cannot be accessed without the sign-up process that includes a drivers license upload; you must own a BCR credit/debit card;
Sixt	No	Private, it only offers a limited selection of electric cars including the likes of BMW i3 and Nissan Leaf	Both	it requires a validation before renting which includes a passport, a drivers license and a selfie; does not have a minimum fare, 0.32 euro per minute for 200km and every extra km is 0.39 euro; offers pre-set hourly or daily packages
Turo	No	Public, it offers a variety of options going from low consumption to high-end super cars	Both	it requires profile validation including a selfie, drivers license, phone number and credit/debit card; it charges per day basis; in-app chat; review system

Table 2.1: Analysis of existing car sharing applications

the application regardless of their favored mobile device, which results in improving the overall user experience and increasing the number of potential users.

Another interesting fact is the difference in the variety of choices when comparing private fleet applications with peer-to-peer ones. Private fleet-based car sharing services typically maintain a restricted selection of company-owned or leased vehicles. While these services may provide certain benefits, such as standard maintenance and dedicated customer service or electric cars, the vehicle options available to users are relatively limited. This can be a significant inconvenience, especially when users have specific needs or preferences, such as the need for a larger vehicle to accommodate a group, a luxury car for a special occasion, or an eco-friendly vehicle to correspond with their sustainability objectives. In contrast, peer-to-peer car sharing platforms connect vehicle owners directly with renters, resulting in a vast and diverse inventory of cars. This wide selection is driven by the fact that individual owners have different vehicle types, models, sizes, and features. Moreover, the wide selection of vehicles allows users to explore and experience different car models without the commitment of long-term ownership or traditional car rental agreements, which adds a note of excitement on renter's note.

Last thing to look at is the availability of the sharing services across the country. Private fleet-based car sharing services typically establish themselves in densely populated urban areas, where there is a high demand. This results in limited availability in smaller cities, so individuals living outside these urban centers may find it challenging to access or benefit from this type of service. On the other hand, peer-to-peer car-sharing platforms have the potential to provide services in almost any location, because there is no limitation of where the rent might happen, as long as the owner can offer the promised services. It can be seen as an advantage since it allows vehicle owners from various regions to monetize their idle assets and contribute to the local economy. In regions where private fleet-based services might not have a presence, peer-to-peer car sharing provides an opportunity for individuals to actively participate in the sharing economy and generate income from their vehicles.

Chapter 3

Mobile Technology and its Role in Peer-to-Peer Car Sharing

We are indeed living in an era where technology has advanced to a remarkable extent, to the point where owning a supercomputer in our pockets has become commonplace. With the rapid progression of mobile devices, such as smartphones and tablets, we now have access to an unprecedented level of computing power and capabilities right at our fingertips.

Recent research indicates that the daily use of mobile devices has consistently increased over the years, with individuals spending a significant portion of their day interacting with these small displays. In order to provide a more precise analysis, mobile utilization increased by 6 minutes per day in 2022, and projections for 2023 indicate a similar or even greater increase. Upon further examination of the analysis conducted by "Insider Intelligence," this discussion must acknowledge an essential aspect. Individuals appear to favor mobile applications over browsers. As seen in the figure below 3.1, the number of mobile app downloads worldwide is growing significantly every year.

In fact, it is said that "daily time spent with mobile browsers will hold steady this year at 52 minutes, before falling to 51 minutes next year." [6] It is safe to say now that developing an user-friendly and intuitive mobile application is has become critical in the current landscape. Relying solely on a single web application that is expected to function on mobile devices may no longer be sufficient, as users' interest in businesses that do not invest in dedicated mobile applications continues to decline.

To meet the demand of users, there are two distinct methods for delivering mobile applications: native and hybrid. Understanding the advantages and disadvantages of each approach is essential for making informed decisions during the development process and the following sections will present and compare the solutions, highlighting their respective strengths and limitations.

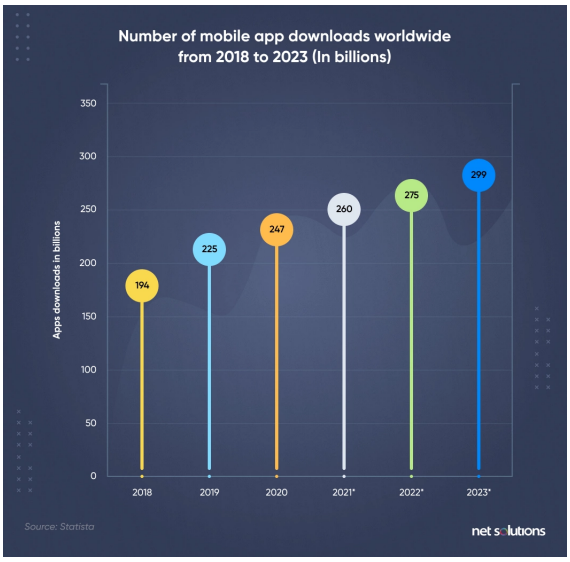

Figure 3.1: Number of apps downloaded worldwide from 2018 to 2023 [18]

By selecting the right mobile solution, peer-to-peer car sharing platforms can take advantage of the benefits of mobile technology, allowing users to rent cars in an easy and comfortable way. With the access to location services and network connection of smartphones, mobile applications can offer features like easy booking, vehicles location tracking and smooth connection between users.

3.1 Overview of Mobile Solutions

The main focus of this chapter will be to provide an overview of the different types of solutions when it comes to development of mobile applications. It is safe to say that the smartphones market keeps growing, as mobile development starts to surpass the traditional software by offering features that desktop or web applications may lack. Mobile phones are equipped with GPS, compass, accelerometer, sensors, and Bluetooth, making them powerful devices for deploying apps.

There is high competition in the mobile industry, where giants as Apple, Samsung, Xiaomi, OnePlus are all ready to invest high amount of money into the development process of new and innovative features for smartphones. As a result, operating systems must keep up the pace by providing constant updates and bug fixing. While there are numerous options for the operating system choice, an article publicated in "International Journal of Applied Engineering and Management" stated that over 99% of the existing smartphones run on either iOS or Android 3.1. As a result, these two operating systems are the primary targets for app development. [8] In order to create an application that is compatible with both iOS and Android we must analyze the two major solutions which are: **native** and **hybrid**.

No.	Mobile Operating System	Percentage
1	**Android**	71.81%
2	**iOS**	27.43%
3	**Samsung**	0.38%
4	**kaiOS**	0.14%
5	**Unknown**	0.14%
6	**Linux**	0.02%

Table 3.1: Mobile Operating System Market Share Worldwide - 2021 [8]

3.1.1 Native Mobile Applications

Native app development is the process of creating an application for a particular operating system, such as Android or iOS. By using this approach, developers can use the maximum performance of the processor and gain access to all the features provided by the OS.

In the context of Android applications, the competition lies between Java and Kotlin as the programming language of choice. Despite Java being one of the most commonly used programming languages, it is remarkable that Kotlin has gained huge popularity and is now referred to as "Google's preferred language" for Android app development, as highlighted in Gabriel Gircenko's article. [9]. In contrast, when evaluating options for iOS devices, Swift and Objective-C are the leading candidates. With a history dating back to the early 1980s, Objective-C has the distinction of being the "veteran" in this comparison. Its compatibility with all versions of iOS and large collection of programming resources contribute to its continuing success. Swift, however, can be linked to "David" in this fight, being only nine years old, but also very powerful due to its purpose-built nature for iOS devices, as it was created by Apple. Swift is currently one of the quickest programming languages, outperforming Objective-C by 2.6 times and Python by an impressive 8.4 times, according to statistics. [21]

There are a number of advantages associated with native mobile applications. First, they provide access to all the features and APIs without requiring additional bridge solutions during development. Second, there are IDEs designed specifically for constructing applications for each platform, such as Xcode for iOS and Android Studio for Android. Lastly, native applications usually display superior performance because they operate directly on the device, without any intermediate layers of code that could affect their efficiency.

Despite its benefits, native mobile application development has a significant disadvantage. Due to the fact that these applications are designed specifically for a single type of device, such as iOS or Android, the code cannot be shared, forcing the

hiring of separate teams of developers to build the application for each platform, resulting in increased costs and efforts for businesses.

3.1.2 Hybrid Mobile Applications

In recent years, hybrid mobile applications have gained significant attention as a viable solution for cross-platform mobile app development. You can think of it as a web application that has the access to a variety of native features and it can be downloaded internally from either Google Play or App store.

The hybrid approach takes the positive aspects of both native and web-based apps and creates a single one that can run on multiple platforms. To do this, a well-designed architecture 3.2 lies behind the application development process and it typically includes three main layers:

- **Native layer.** Depending on the platform (Android or iOS), it is typically written in languages like Java or Swift. It provides access to the hardware and OS features of the device such as camera, sensors, push notifications etc.
- **Web Layer.** This is the core of the application that is built using web technologies such as HTML, CSS, and JavaScript, and it can be developed using frameworks like React, Angular, or Vue.js. At the base, it consists of a WebView that functions as a container in which the web app is displayed.
- **Bridge.** As you can expect from the name of it, a bridge connects the native layer to the web layer, allowing them to interact. It provides a set of APIs that the web app can use to call native functions and access device-specific functionalities.

Figure 3.2: Hybrid applications architecture [17]

One of the primary benefits of the hybrid approach is that a team of developers can create a single application for both Android and iOS platforms, as the code base is often identical. This technique is known for its cost and time efficiency, making it a preferred choice for companies, including industry giants such as Uber, Google, and Twitter.

Even though it might sound as the 'go-to' choice when planning on building a new mobile application, there are some drawbacks that are worth mentioning. One significant concern is the potential for decreased performance in comparison to native applications, since hybrid apps rely on web technologies which may not perform as efficiently as the platform-specific languages used in native ones. In addition, issues may arise during the development process with APIs or modules that function effectively on Android but have limitations on iOS or vice versa.

3.2 Comparison of Native and Hybrid Mobile Solutions

Prior to selecting the optimal technology for this peer-to-peer car sharing application, there is this special sub chapter that compares the two mobile solutions in more detail. By highlighting the key characteristics of native and hybrid solutions, we can make an informed decision regarding the most suitable technology for this project.

	Native Mobile Apps	**Hybrid Mobile Apps**
Development	Platform-specific (Android, iOS)	Single codebase for multiple platforms
Performance	Excellent performance and responsiveness	Slightly lower performance compared to native apps
Device Access	Full access to device features and APIs	Limited access to certain device features
User Experience	Native look and feel, seamless integration	Can achieve native-like experience
Development Time/Cost	Longer development time, higher cost	Faster development time, cost-effective
Maintenance	Separate maintenance for each platform	Unified maintenance for multiple platforms
Offline Support	Can work without an internet connection	Dependent on internet connection for functionality
Updates	Independent updates for each platform	Simultaneous updates for multiple platforms

Table 3.2: Native vs. Hybrid Mobile Solutions

3.3 Choosing the Appropriate Mobile Solution

The choice of the appropriate mobile solution was made after careful consideration and answering to some several key questions. These questions may also help other developers who find themselves into the same dilemma of not knowing what approach is the best for their purposes. I will start by listing the answers to those questions, providing detailed arguments for each choice. Moreover, I will share my personal preferences on the subject and present some of the features that influenced my decision.

Am I constrained by development time and budget?
A peer-to-peer car sharing application is very important to be available on both platforms in order to satisfy all types of users and to increase popularity and accessibility. Native apps typically require separate development efforts for each platform, resulting in longer development time and higher costs. Considering that I am still a student without a team that can back me up or a substantial budget, I need a method that can suit my possibilities. Therefore, Hybrid apps, such as those developed using the Ionic framework, allow for faster development with a single codebase. This reduces costs and simplifies maintenance, making it a suitable choice for me.

Do I need the application to work without an internet connection?
I think that both options would work for this answer, since a peer-to-peer car sharing application does not necessarily need to work offline. A user must have internet connection in order to search for a car, request a car or just speaking with the owner. These things cannot go offline, since there can be changes while the connection is lost, so the outcome is different from the desired one.

How important is performance for our application?
Performance can be a very important aspect when it comes to developing a mobile application. It is clear that native apps offer better performances and responsiveness than what hybrid apps may exhibit. However, a peer-to-peer car sharing application does not require a near-zero response time, as is the case with games, for example, where half a second can make the difference between victory and defeat. The best example to prove my point is to use the Uber app as a reference. Its functionalities are similar to what can be found on a peer-to-peer car sharing app and they use the hybrid solution for the development process. So, the point may go to Hybrid from here, too.

How important is achieving a native look and feel?
With the UI components that Ionic provides in their library, I think that Hybrid apps can also achieve similar user experience to a native app. In this way, I do not think of any reason why one is better than the other.

How important is to have full access to device features and APIs?
The feature that must be accessible for a peer-to-peer car sharing mobile applcation are the following: keyboard, camera (it is required to be able to take pictures of the car and validation purposes) and location services (it is required to display the desired cars available within user's area). These can be easily accessed on both native and hybrid (with the help of capacitor bridge), so the answer may not impact so much the final decision.

It is quite clear that the scale tilts towards hybrid apps, due to the constraints of time and budget, as well as the performance of the application, which heavily influence the answers provided above.

Furthermore, once I have chosen to develop the application using a hybrid solution, I can confidently say that Ionic is a personal preference, as it is built on the Angular framework. The experience I have gained so far working with Angular has made the development process more enjoyable and efficient, as I am already familiar with many of its functionalities. Additionally, Angular is the go-to solution for developing complex applications like this peer-to-peer car sharing app.

3.4 Future Trends and Considerations

As technology continues to evolve at a rapid pace, it is important to consider future trends and potential advancements that may impact the peer-to-peer car sharing industry. I have gathered two of the most relevant ways in which this peer-to-peer car sharing app could extend in the future to improve the user experience and information security, namely: Blockchain and IoT.

Blockchain technology can provide a secure and transparent platform for car sharing. By using blockchain, important information such as vehicle history, insurance records, and driver licenses can be stored securely and verified. This ensures the reliability and integrity of the data, promoting trust among users. [2] Furthermore, car rental transactions could be implemented directly through blockchain. The smart contract can automatically verify the availability of funds in the renter's digital wallet. Once the rental period is completed, the smart contract can release the payment from the renter's wallet to the car owner's wallet. Not only does this method eliminate the need for intermediaries (typically a bank), but it also elimi-

nates the transaction fees associated with traditional banking systems.

The integration of **IoT (Internet of Things)** in peer-to-peer car sharing applications can enable a semaless connection between the mobile application and the actual vehcicle, making the experience of car rental more convenient and user-friendly. Through IoT devices and sensors installed in the vehicles, users can remotely access and control various functionalities such as unlocking and locking the car, adjusting temperature settings, and monitoring fuel levels.

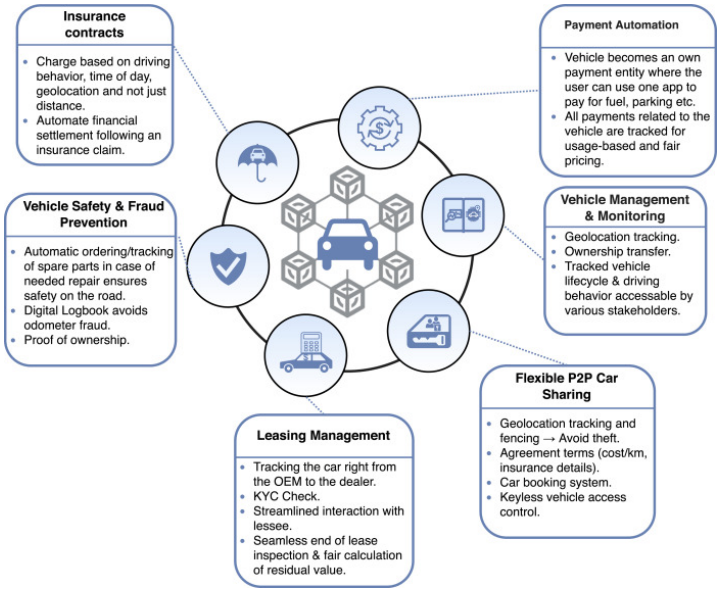

Figure 3.3: Overview of use cases enabled by the conceptual design. [2]

Overall, these features would open new possibilities for this peer-to-peer car sharing application, by staying up-to-date in terms of technology and exceeding user expectations.

Chapter 4

Implementing the Peer-to-peer Car Sharing Application

This chapter marks the point where we dive into the implementation details of our peer-to-peer vehicle sharing mobile application, following a thorough examination of the related work and technologies used in the development process. Based on the knowledge acquired in previous chapters, we can now shift our attention to the application's actual creation. This chapter seeks to provide a comprehensive overview of the app's construction by describing its architecture, design considerations, development methodologies, and technical obstacles encountered during the implementation phase. In addition, it offers the opportunity to demonstrate how my vision has been materialized and how the identified requirements and specifications have been translated into functional components and features.

4.1 Requirements

The purpose of this theoretical subchapter is to focus on the functional (FR) and non-functional (NFR) requirements for the development of this peer-to-peer car sharing mobile application. The functional requirements are the key features that the application must fulfill, while the non-functional requirements explain how the system should perform from different points of view including performance, security, scalability and accessibility.

4.1.1 Functional requirements

- **FR0: User Registration and Profile Management**
  Users should be able to register an account and provide essential details such as name, email address, phone number, and profile picture.
  The application should allow users to edit and validate their profiles by providing a valid driving license document.

- **FR1: Car Listing and Management**
  Users should have the ability to add cars to the platform for rental purposes. This includes providing detailed information such as car make, model, year, transmission type, fuel type, and photos.
  The application should allow car owners to manage their listed cars, including editing information, updating availability, and removing cars from the platform when necessary.

- **FR2: Car Search and Filtering**
  Users should be able to search for available cars based on their location and other filters, such as car model, year, transmission type, and other specifications.
  The application should provide users with a list of relevant car options that meet their search criteria, displaying important details and images.

- **FR3: Reservation and Booking Management**
  Users should be able to reserve a specific car for a particular period, creating a booking request that is sent to the car owner.
  Car owners should have the ability to accept or cancel booking requests within a specified time frame.
  The application should provide users with a booking confirmation, including details of the reserved car and rental period.

- **FR4: In-App Messaging**
  Users should be able to communicate with car owners through an in-app messaging system to discuss rental details, coordinate pickup and drop-off locations, and ask any questions related to the rental.
  The application should provide a convenient and secure chat interface to facilitate communication between users.

- **FR5: Review and Rating System**
  After completing a rental, users should have the option to leave a review and rating for the car and owner, reflecting their experience.
  The application should allow users to provide feedback, helping other users make informed decisions and maintain a reliable and trustworthy community.

4.1.2 Use cases

In the last section, the functional requirements of the 'Peer-to-peer car shaing' application were clearly stated in order to understand their general behaviour. Next, there is an use-case diagram 4.1 which illustrates how these functional requirements are related to user interaction.

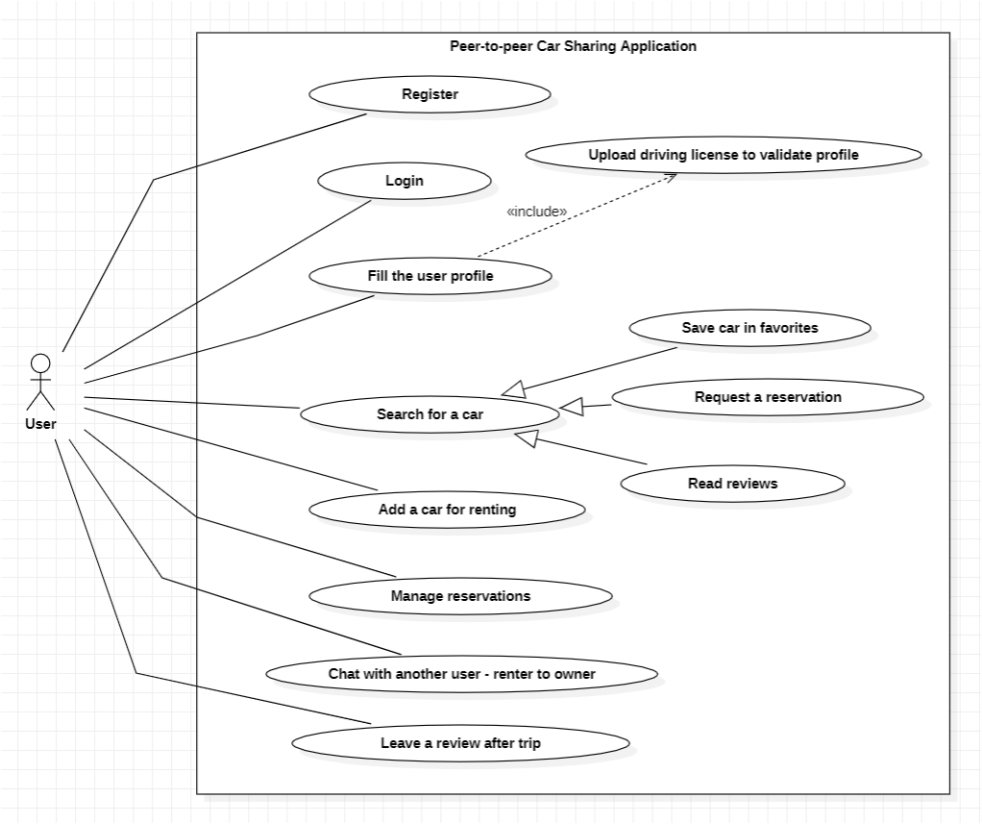

Figure 4.1: Use-case diagram representing application's functionalities

When referring to use cases in his book [14], Jacquie Barker asks the following question "When users interact with the system for a particular purpose, what is their expectation as to the desired outcome?". I refer to this question in order to introduce the use case description tables that will answer it, by presenting the user expectations according to some of the key functionalities.

In the following table, it is presented the 'Add car' functionality use case:

Name	Add a car
Actor	User
Description	User adds a car for rental
Pre-conditions	User is on 'My Vehicles' page
Flow	1. User clicks on '+' button 2. System presents the modal which contains inputs for car details and images 3. User completes all required inputs and press save 4.1. If the inputs are correctly filled, the system closes the modal and a new car is added in the list of cars 4.2. If the inputs are not correctly filled, a toast will be presented informing the user of the errors
Post-conditions	The application adds the car into the database

Table 4.1: 'Add car' functionality use case description

In the following table, it is presented the 'Manage Reservations' functionality use case:

Name	Manage Reservations
Actor	User
Description	User accepts a reservation request
Pre-conditions	User is on 'My Car Requests' page
Flow	1. User clicks on 'Confirm' button 2. System updates the status to 'ACCEPTED' 3. System automatically declines all the requests for that specific car in the same period with the one accepted 4. User can see the updated interface of the request
Post-conditions	The application completes the confirmation flow

Table 4.2: 'Manage reservations' functionality use case description

By automatically declining the requests that overlap with the one that the user accepts, it is ensured that a car cannot have multiple reservations during the same period of time, thus avoiding unpleasant situations where two users are notified that they will rent the same car for the same interval, which is indeed impossible.

In the following table, it is presented the 'Leave a review after trip' functionality use case:

Name	Manage Reservations
Actor	User
Description	User leaves a review after trip
Pre-conditions	User is on 'My Trips' page
Flow	1. User clicks on 'Leave a review' button, after the trip is completed 2. System opens a modal which contains inputs for rating and text 3. User completes the inputs and clicks submit 4. System stores the review and computes the new rating for the car 5, User can see a toast informing that the review was saved successfully and the 'Leave a review' button will disappear
Post-conditions	The review is stored into the database

Table 4.3: 'Leave a review' functionality use case description

4.1.3 Non-Functional requirements

- **NFR0: Security**
  The application must prioritize the security of user data, implementing appropriate encryption techniques and secure authentication methods.
  User privacy should be respected, with clear policies regarding the storage and handling of personal information.

- **NFR1: Performance and Scalability**
  The application should be designed to handle a large number of users and cars, ensuring smooth performance even during peak times.
  It should support scalability to accommodate potential future growth and increasing demand.

- **NFR2: User-Friendly Interface**
  The application should feature an intuitive and user-friendly interface, making it easy for users to navigate through different sections and perform desired actions.
  Clear and concise instructions should be provided to guide users through the various steps of the rental process.

- **NFR3: Compatibility and Cross-Platform Support:**
  The application should be compatible with popular mobile operating systems (iOS, Android) to cater to a broader user base.
  It should be optimized for various screen sizes and resolutions, ensuring a consistent and visually appealing experience across different devices.

4.2 Application specification

My aim is to provide a secure and easy to use platform for users to share their cars with others for rental purposes, enabling efficient and cost-effective transportation options. In the following lines, the key features of the application will be presented in order to gain a deeper understanding of how the main flow was designed.

Starting with the basics, there is a rigorous authentication/registration process. Users are able to create an account by providing their first name, last name, email address and password. In order to gain access to all the features of the application, they will be required to complete user details with a profile picture, phone number for direct communication with other users and birthday. The process will be ready after a driving license validation, where users are asked to upload a photo of their driving license from an angle that facilitates reading information.

After completing the 'Personal details' section, users are able to search for available cars based on their location and other filters, as brand, model, transmission etc. Once a suitable car is found, the user can send a rental request to the owner. The request can be accepted or declined by the owner. If the owner does not respond the start date of the request, it will automatically be marked as declined.

The application allows car owners to upload their vehicles to the platform for sharing. Owners must provide details about their cars, including make, model, year, power, number of seats, transmission and fuel consumption. Also, more important details as rental rates and pickup location cannot be skipped. Visual appeal is essential for attracting prospective renters. In regard to this, the application enables car owners to upload images in order to present their vehicles in the best possible way, emphasizing their features and condition. The presence of high-quality images can build trust and confidence, thereby increasing the chances of successful rentals.

By recognizing the importance of effective communication, the application includes an in-app chat function to facilitate secure and efficient interactions between renters and owners. This feature provides both parties with a dedicated channel for real-time communication, allowing them to discuss rental terms, negotiate pricing, and coordinate pickup and drop-off details. By providing a direct communication channel, the in-app chat feature eliminates the need for external messaging platforms, thereby assuring a seamless and integrated experience within the application.

In order to allow users to provide feedback and ratings based on their experience with rented cars, the application has a review system, which is a very important asset for a car sharing app. After completing the trip, renters can evaluate the overall condition and functionality of the vehicle, as well as the communication and cooperation with the owner. These reviews and ratings serve as a valuable resource for

other users who are considering renting a specific vehicle. They provide an objective perspective on the vehicle's performance and the overall renting experience. Renters can rely on these evaluations to make informed decisions, ensuring that they select a vehicle that meets their needs and expectations.

The application can handle large number of potential users and it successfully addresses the security concerns, because of the data encryption and prevention of unauthorized access. By creating an intuitive design that has the correct balance of color and professionalism, it is ensured that users can easily navigate throughout the app and perform different actions. Finally, the application is compatible with both iOS and Android, providing great accessibility to all types of smartphones.

4.3 Application architecture

The architecture of a peer-to-peer car sharing mobile application involves several components working together to provide a seamless and efficient user experience. In this case, the application was developed using the Ionic framework for the frontend, while Firebase was utilized as a backend substitute. This choice was made due to the inherent advantages of Firebase, such as its speed, security, and reliability. It should be mentioned that a small part of the logic which involves the driving license validation is separately implemented in Python language and accessed with the help of a HTTP POST request.

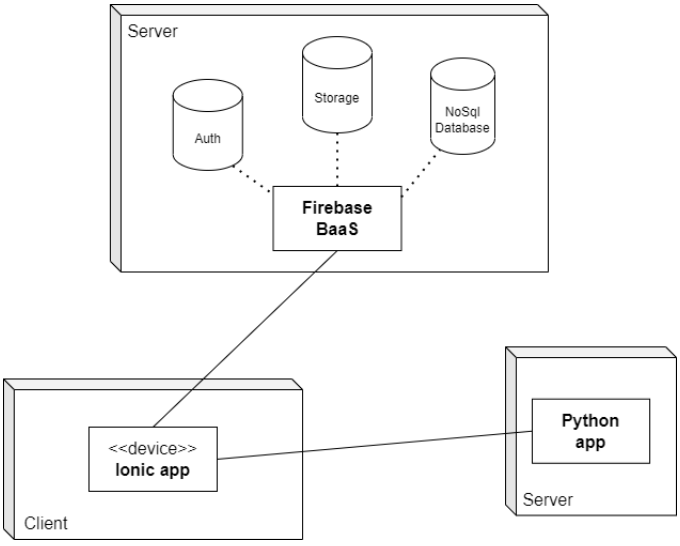

Figure 4.2: Architecture diagram

4.3.1 Client-side

At the core of the application is the Ionic framework, which provides a platform-independent method for developing mobile applications with web technologies such as HTML, CSS, and JavaScript. Ionic enables the creation of a single Angular codebase that can be deployed on multiple platforms, including iOS and Android, thereby assuring a consistent user experience. The architecture of this client-side application is organized into services, sub-components, and pages, which provide a structured approach to the development process.

Services

The 'services' folder contains all the essential components responsible for connecting and interacting with Firebase and our Python backend application. These services act as intermediaries between the front-end and the backend, enabling seamless communication and data exchange. Each service encapsulates a specific functionality, such as user authentication, vehicle management, messages handling for chats, review management and trip administration. By separating these functionalities into services, we ensure integrity and maintainability of the codebase.

Sub-components

All the reusable components that are utilized throughout the application are implemented into the 'sub-components' directory. These components include modals and items that are utilized in various sections of the mobile app. By encapsulating these reusable elements, we promote code reusability, reduce redundancy, and facilitate consistent user experience across different pages.

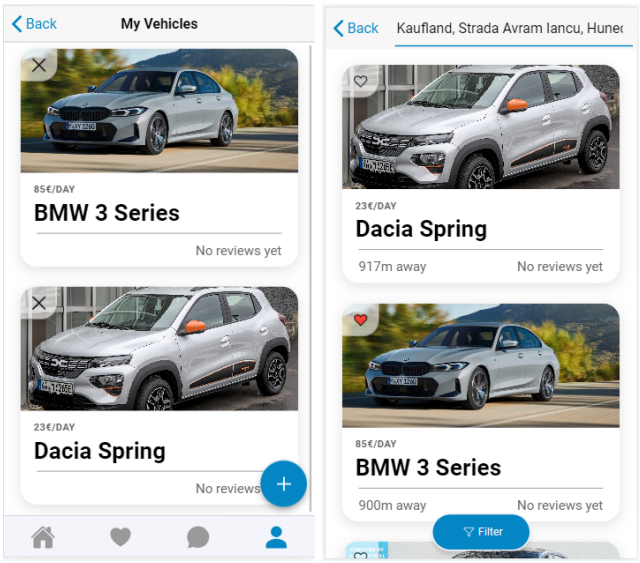

Figure 4.3: Car item displayed in different sections of the application

To exemplify the notion of reusability, let's take the sub-component 'large-car-item' as example. 4.3 This sub-component has been programmed to possess versatility and adaptability to various contexts within the application. Its capability to transform in different visual representations is due to the 'showFavoriteButton' property which can be set to true/false when referring the sub-component into a page. By setting this property to true, the functionality of the 'X' button, initially used for removing the car in the MyVehicles section, transforms into a 'heart' button. This transition occurs when the sub-component is used in the search page or favorites page, where the 'heart' button allows users to add the car to their favorites list. Also, a new label is displayed providing the user with information on the distance between the car and their current location or the one entered in the input.

Pages

The pages are the main navigational units of this mobile application. Each page represents a distinct user interface and serves as an entry point for specific functionalities. Within the pages folder, we have organized child pages under their respective parent pages to ensure a logical and hierarchical structure. For instance, the MyAccount page contains child pages such as MyProfile, MyVehicles, and MyTrips, providing an intuitive navigation flow. The pages have the responsibility of presenting relevant information and managing user interactions with the system. They utilize the available services and sub-components to retrieve data, handle states, and display information in a user-friendly format.

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="home">
      <ion-icon name="home"></ion-icon>
    </ion-tab-button>
    <ion-tab-button tab="favorites">
      <ion-icon name="heart-sharp"></ion-icon>
    </ion-tab-button>
    <ion-tab-button tab="chats">
      <ion-icon name="chatbubble-sharp"></ion-icon>
    </ion-tab-button>
    <ion-tab-button tab="account">
      <ion-icon name="person-sharp"></ion-icon>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

Figure 4.4: 'ion-tabs' + 'ion-tab-group' HTML template of the application

It is worth to mention that the structure of the application is organized using the 'ion-tabs' component, together with 'ion-tab-group' component, both offered by Ionic. 4.4 These two can be also used separately, but they usually work together in order to create a user-friendly tab-based application that behaves like a native one. [12] 'ion-tabs' component serves as a container for the four main tabs of this application: Home, Favorites, Chats, and MyAccount. Each tab represents a distinct section of the application with its own functionality and content. 4.5

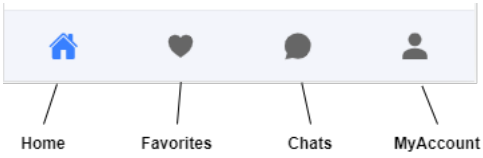

Figure 4.5: Navigation container for the four main tabs of the application

By using 'ion-tabs', the app allows users to switch between sections and access the corresponding features and information. For example, the Home tab allows users to search for available cars, the Favorites tab displays a list of cars stored in the favorites list, the Chats tab provides a platform for direct communication between users, while the MyAccount tab grants access to a range of user-specific pages.

Beyond the implementation of the 'ion-tabs' component, the application relies on the 'router-outlet' to dynamically render the content of these pages. The 'router-outlet' acts as a placeholder where Angular's router injects the appropriate component based on the current route. This enables the implementation of dynamic navigation and content presentation, guaranteeing a smooth user experience. In addition, the router-outlet is used with the auth guard to implement access restrictions. For example, if a user is not authenticated, they will be automatically redirected to the login page.

4.3.2 Server-side

In the context of developing a mobile application, the server-side implementation plays a crucial role in ensuring the seamless operation and efficient management of the application's data. The server-side can be either a traditional back-end server which is an application that can be written in a range of languages (Java, PHP, Python, Go etc.) or a back-end as a service (Baas) platform. The most popular providers when it comes to Baas are: Google (Firebase), Amazon (AWS Amplify), built.io etc.. [16]

The choice was to use Firebase Back-end as a service for this project, since it offers a wide range of useful features that can effectively handle the server-side requirements of this application. Firebase has some real advantages over traditional back-end servers and it is worth mentioning some of them:

- **Scalability.** The cloud-based architecture of Firebase automatically scales resources to meet the demand of an application. It manages the allocation of computational resources and traffic distribution, ensuring optimal performance even during peak usage periods.

- **Rapid Development.** By providing a pre-configured and ready-to-use backend infrastructure, Firebase enables developers to focus on constructing core application functionality. This allows for quicker development by eliminating the need for setting up servers, configuring databases, and managing complex infrastructure.

- **Real-time Updates.** Firebase's real-time database and synchronization capabilities allow for synchronized real-time data changes across multiple clients. This ensures that any changes made to data on one device are immediately reflected on all other connected devices, ensuring a consistent and up-to-date user experience.

Let's focus now on the Firebase features that were used in the development process of this application: Authentication, Real-time Database and Storage.

Authentication

Authentication represents a very important component in any application whether it is mobile or web, since it ensures secure access and data privacy. While Firebase offers a wide range of sign-in methods as email/password, phone number, Google account, Twitter account, Facebook account and many more, this project focuses on the implementation of the classical email/password authentication method. To execute this method, the program prompts the user to input their email, first name, last name and password as part of the registration procedure. Firebase Authentication securely stores the provided credentials by hashing the password and validates them during subsequent login attempts using a secure authentication mechanism. An identity token is generated upon successful email authentication and is subsequently included in the header of each request made to the backend API. This token follows industry-standard protocols such as JSON Web Tokens (JWT), enabling secure and stateless communication between the client and server.

Firestore Database

The Firestore Database is a NoSQL database hosted on the cloud that provides real-time synchronization and data persistence across multiple clients in real time. The system facilitates smooth incorporation with mobile and web apps, providing a powerful backend infrastructure. NoSql database employs a hierarchical structure known as a JSON tree. The data is organized into a collection of key-value pairs, with each key uniquely identifying a specific node within the tree. The keys can be used to navigate and retrieve data efficiently. Please refer to the attached diagram for a visual representation of the NoSQL database structure used in the 'users' collection. 4.6

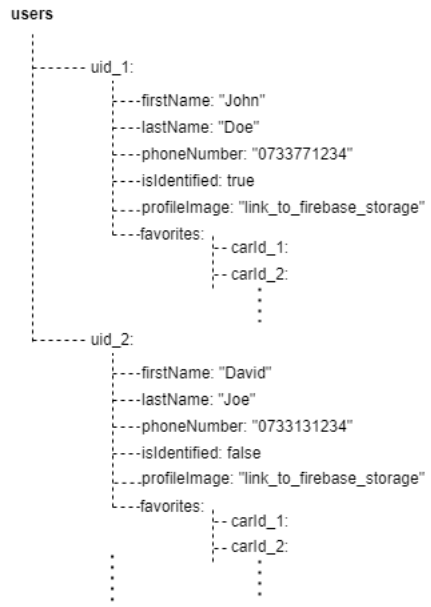

Figure 4.6: NoSql 'users' collection representation

The collections used for this peer-to-peer car sharing application are the following:

- **'users'.** This collection stores user data as 'firstName', 'lastName', 'phoneNumber', 'birthday', 'profileImage' and a boolean 'isIdentified' to decide if the profile validation using the driving license was successfully done or not. Additionally, an extended collection named 'favorites' is user in order to save the IDs of the selected favorite cars for each user.

- **'cars'.** It contains car details such as 'brand', 'model', 'transmission', 'location-Geohash' and many other details on order to give users as much information as possible about the car they are going to rent. Also there is an extended collection named 'reviews' which is used to store the reviews written by users after trips completed with the specific car.

- **'chatRooms'.** This collection is responsible for managing the various chat rooms created within the application. It stores information such as 'members', 'createdAt' and 'updatedAt'.

- **'chats'.** It is used to store individual chat messages exchanged between users within the chat rooms. The representation would be chatRoomId -> 'messages' -> messageId -> {'sender', 'createdAt', 'message'}.

- **'reservations'.** It manages the reservations made by users for specific cars. It stores details such as the user making the reservation, the car reserved, the start and end dates of the reservation and the status.

It is required to install the Firestore SDK to be able to work with the Firestore Database in Angular. Using the AngularFirestore class, it is pretty straightforward to establish a connection with the database and to retrieve the desired data from there. By listening to changes with the help of 'valueChanges()' method provided in AngularFirestore, an observable that emits data is returned whenever the values in the specified collections are modified.

Storage

Firebase Storage is a cloud-based solution to save files, images in this case, in a secure manner. The architecture consists of multiple data centers strategically distributed across different geographical regions. This ensures that images are stored and retrieved efficiently, irrespective of the user's location.

When a user uploads an image via a mobile application, Firebase Storage transparently manages the entire process. A secure HTTPS connection is used to transmit the image from the user's device to the Firebase servers.

Within this peer-to-peer car sharing application, there exist two distinct folders: "carImages" and "userImages." The purpose of the "carImages" folder is to house images of available cars for online showcasing and rental purposes. The file path for these images follows the structure: "carImages/userId/carId/" with additional images denoted by "image1," "image2," and so on. On the other hand, the "profileImages" folder serves as a storage location for user profile pictures. The path for these images is relatively straightforward: "profileImages/userId" with the userId serving as the name of the profile photo.

4.4 Design

In current era, the design and user interface are very important aspects when it comes to development of a mobile application. Because of a large number of applications in each domain, users are more attracted to the ones that are intuitive and are visually appealing. Next lines of this section will be about how SCSS (Sassy Cascading Style Sheets) and Ionic UI components helped in the development of a professional good looking interface.

4.4.1 Sassy Cascading Style Sheets (SCSS)

SCSS enables the creation of reusable classes that can be utilized across different items or components within the application. This is a very useful feature that simplifies the developer's life quite a lot. Moreover, it is a very good practice to decide on a set of visual properties, such as color palette and text font, before the

actual implementation of the application, set it into the global.scss file and then use the declared values whenever needed throughout the project. This approach offers flexibility as well, since any changes made to the global.scss file will automatically propagate throughout the entire app. For instance, by modifying a color code in the global.scss declaration, the colors across the application can be easily updated. Here is an example of how it is implemented in the application:

```
:root {
  --primary-color: #0486C4;
  --secondary-color: #f3f5fb;
  --primary-text: Roboto, serif;
}
```

```
p{
  font-size: 20px;
  font-weight: 700;
  color: var(--primary-color);
}
```

Figure 4.7: Class declaration is presented on the left, while its utilization is presented on the right

In this car sharing app, the combination of the colors #0486C4 (light blue) and #f3f5fb (white) adds a fresh and attractive appearance without compromising professionalism. These colors are used consistently throughout the app, including the login page, where buttons and texts are designed to match with the color of the car in the background. 4.8 The transitions are also present in this login section of the app (when clicking on [Let's go], text fades away on the right, the inputs pop out form the bottom of the screen and the background slowly blurs), demonstrating the attention to details and dedication to providing users with a positive impression.

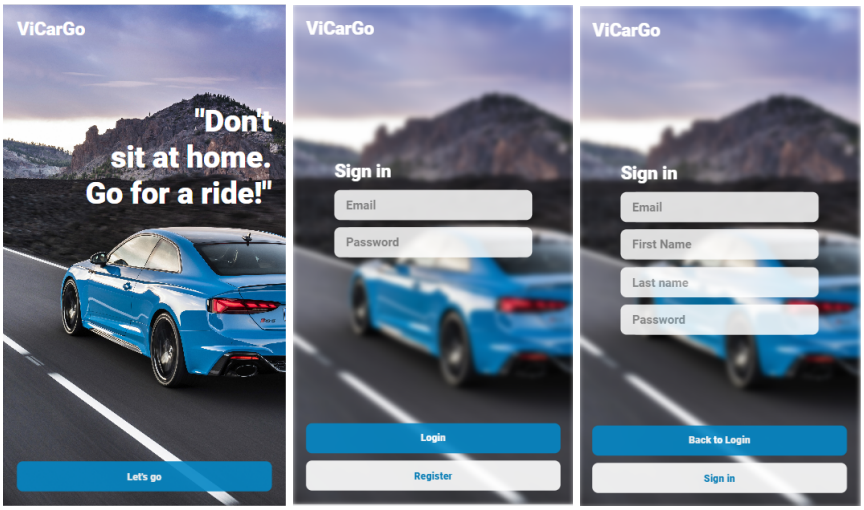

Figure 4.8: Representation of login section design

4.4.2 Ionic UI components

Ionic offers a free to use UI component library filled with a wide range of pre-built and ready-to-use elements which are available for the development process. There is this documentation on Ionic website [12] where every component is cut open and

analyzed in order to give developers a full understanding of how does it work. By integrating them into the mobile application, developers can benefit of consistency, responsiveness, customization and a native-look of the app.

It is worth to mention some of the Ionic UI components used throughout the application: ion-content, ion-item, ion-card, ion-icon, ion-label and many more. In addition, I would like to speak about three of them which deserves separate attention: ion-icon, ion-datetime and ion-skeleton-text.

ion-icon is a very useful tool when it comes to displaying icons in the app. Over 1000 icons are ready to be chosen from the Ionicons library and the usage is efortless. A name of the icon is required and additionally size and color can also be set. [12]

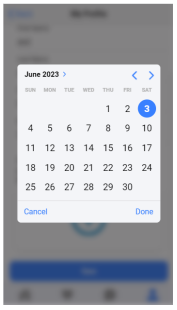

Figure 4.9: Date picker modal in action

ion-datetime is a nice looking date picker modal that it is present in different places of the application, including birthday input in user profile section or start/end dates for car reservation. This version 4.9 is also customized in order to provide a better looking representation.

ion-skeleton-text is used as a replacement when the application needs to load data. It is a good practice to use this type of loading indicator, since it adds a touch of professionalism. For example, a skeleton card built from ion-skeleton-text elements was built as a replacement for the large-car-item while loading the data from Firebase database:

Figure 4.10: Skeleton card built using ion-skeleton-text

4.5 Special functionalities

In this subchapter, we delve into the special functionalities that set our application apart. Two of the most challenging and impactful features include driving license validation and geohashing-based car filtering. The advanced technology employed enables seamless validation of driving licenses, ensuring only authorized individuals can access certain services. Additionally, the geohashing algorithm efficiently filters cars based on location, enabling users to find the nearest available vehicles with ease.

4.5.1 Driving license validation

The implementation of a driving license validation function into a peer-to-peer car sharing mobile application is fundamental. This feature requests an image of users' document and automatically scan it in order to improve the safety level of the application and diminishing liability concerns. By verifying the credentials of users, the application creates a trustworthy environment, thereby boosting user confidence and facilitating agreements between car owners and renters.

In this process, I used the help of text detection feature from the Cloud Vision API, which is a cutting-edge image recognition service offered by Google, designed to analyze and understand the content of images. Using complex machine learning algorithms, this API can accurately identify and categorize objects, detect faces, and understand text within images and enables developers to incorporate powerful image analysis capabilities into their applications effortlessly.

During the implementation of the driving license validation feature using Google Cloud Vision API (OCR) for Romanian driving licenses, I encountered a significant challenge. The driving license contains some tiny text colored in pink between the black text that was needed to be extracted for validation purposes, as it can be seen in the image below in the yellow highlighted area of the document. 4.11 Initially, I struggled to find a consistent pattern in the output text generated by the API, because of those extra words that were coming from the background pink text, which almost made me abandon the idea of implementing this feature altogether.

After several hours of intense brainstorming, I came up with the idea of applying a grayscale filter to convert the colors into black and white. The intention behind this approach was to transform the unwanted ”pink” text into white, while preserving the essential ”black” text required for validation. However, despite my efforts, this modification did not produce the desired results. One more final adjustment, of transforming the white color to a purer white and intensifying the blacks to a darkened tone, was needed in order to prepare the image for text extraction and to

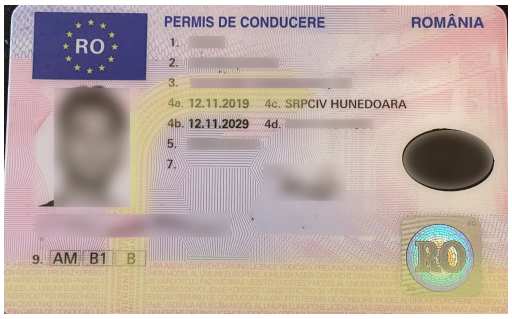

Figure 4.11: Romanian driving license template

obtain the desired output result. This simple yet crucial modification proved to be the turning point in this process of text extraction.

Leveraging the power of regular expression (regex) methods, I successfully implemented this automated validation method. This method validates the profile data entered by the user, using a separate Python endpoint to which the user submits the driving license image and their profile details.

The first requirement of this validation process is to ensure that the image contains the text ”PERMIS DE CONDUCERE”. If this condition is not met, the status is immediately returned as ”invalid”, in order to inform the user that he may uploaded an image of another document, not the driving license. However, if the text is present, the system proceeds to check the consistency of the first name, last name, and birthday provided by the user. For improving accuracy, the validation process for first and last name involves the removal of diacritics and the conversion of text to uppercase. Additionally, the preferred format specified from the frontend is utilized for verifying the validity of birthday. 4.12

```
# Validate first name
first_name_match = re.search(r"2\.\s(.*?)\s3\.", ocr_result)
if first_name_match:
    first_name = first_name_match.group(1)
    first_name = unicodedata.normalize("NFKD", first_name)\
        .encode("ASCII", "ignore").decode("utf-8")
    print("First Name:", first_name)
    input_first_name = unicodedata.normalize("NFKD", input_first_name)\
        .encode("ASCII", "ignore").decode("utf-8")
    if first_name.upper() != input_first_name.upper():
        return 'retry'
```

```
# Validate birthday
birthday_match = re.search(r"3\.\s(.*?)\s", ocr_result)
if birthday_match:
    birthday_str = birthday_match.group(1)
    birthday = datetime.strptime(birthday_str, "%d.%m.%Y")
    birthday_formatted = birthday.strftime("%Y-%m-%d")
    print("Birthday:", birthday_formatted)
    if birthday_formatted != input_birthday:
        return 'retry'
```

Figure 4.12: First name and birthday validation methods

The final verification step involves ensuring that the current date falls within the validity period specified on the driving license, i.e., between the date the license was declared valid and its expiration date. If any of these four requirements fail, the system returns a status of ”retry”, meaning that the system recognises the driving license, but there could be minor issues related to image processing or potential misspellings in the user input. In this case, the application informs users to double check the inputs and to take or upload a new photo from a different angle. Conversely, if all the requirements are successfully met, the system deems the driving

license as "valid" and the application give him access to all the available functionalities.

4.5.2 Geohashing filtering and auto-fill location input

The easiest way of displaying a list of cars filtered by their location is to retrieve them all from the Firebase Database and then calculate their distance from a specific point. This calculation is performed using the Haversine formula, which relies on the latitude and longitude values stored within the car entity. Obviously, this is not what experienced programmers are willing to do, due to its consumption of a significant amount of memory and deceleration of the application.

One valid approach to achieve the desired outcome is to use Geohashing, which is a technique used to encode geographic coordinates into a short alphanumeric string, representing a rectangular area on the Earth's surface. 4.13 This feature offers a practical method for organizing and retrieving geographical locations by their relative proximity. Geohashes are produced through the process of partitioning the globe into a series of cells and distributing a distinct code to each individual cell. The precision of the location represented by a geohash increases proportionally with the length of the string. [7]

Figure 4.13: Earth divided in cells using Geohashing [7]

If we zoom in, the selected bucket will split into 16 more rectangular areas and this happens recursively for every new letter added to the string. For example, consider the coordinates (41.390205 lat, 2.154007 long), which correspond to Barcelona, Spain. By converting these coordinates into a geohash, we can obtain a compact representation of the location. 4.14 Similarly, nearby areas will have geohashes that share a common prefix, making it easy to perform proximity-based searches.

On the programming part, the Angular framework offers a library named **'ngeohash'**, which provides a convenient set of functions and methods to generate

Figure 4.14: Barcelona represented by Geohashing [7]

geohashes and decode them back into coordinates (latitude and longitude). With the help of this methods, the application stores the geohash string into the database for every car entity, as location is a required input in the process of adding a car.

```
getGeoHashRange(longitude: number, latitude: number, radius: number) {
  const latDegreePerKm = 1 / 110.574; // Degrees latitude per kilometer
  const lonDegreePerKm = 1 / (111.320 * Math.cos( x: latitude * Math.PI / 180)); // Degrees longitude per kilometer

  const lowerLat = latitude - (latDegreePerKm * radius);
  const lowerLon = longitude - (lonDegreePerKm * radius);

  const upperLat = latitude + (latDegreePerKm * radius);
  const upperLon = longitude + (lonDegreePerKm * radius);

  const lower = ngeohash.encode(lowerLat, lowerLon);
  const upper = ngeohash.encode(upperLat, upperLon);

  return {
    lower,
    upper
  };
}
```

Figure 4.15: Method used to find upper bound and lower bound geohash strings of desired location

Next, a method named 'getGeoHashRange' 4.15 has been implemented which requires 3 parameters: latitude and longitude of user's location, as well as the specific range within which the desired cars should be found (5 kilometers by default). The purpose of this method is to create a boundary for firebase query 4.16 as it returns the upper bound and lower bound geohash strings of the desired location. In this way, the list returned by firebase contains exclusively cars within the user's selected area, thereby optimizing memory usage by discarding irrelevant entities.

```
return this.firestore.collection( path: 'cars', queryFn: ref => ref
  .where( fieldPath: 'locationGeohash', opStr: '>=', lowerBoundHash)
  .where( fieldPath: 'locationGeohash', opStr: '<=', upperBoundHash))
  .valueChanges();
```

Figure 4.16: Query based on boundary values returned by getGeoHashRange

Having discussed how does the filtering mechanism work, you may now be curious about how the exact location for cars is obtained. Well, to achieve this, there

is this input present in the form where users can add a car for renting. However, it is not just a simple input, as it uses the Google Places Autocomplete functionality, which dynamically presents a list of valid addresses as the user types letters into the input box. Once the user selects the desired address, Google provides the precise location coordinates in terms of latitude and longitude. These coordinates are then used to generate a geohash, which is stored into car entity for future filtering purposes.

4.6 Testing and validation

Testing and validation are crucial stages in the development of an application. This process ensures that the application functions as expected and can identify and address potential issues before the final release.

In order to test the presented 'Peer-to-peer car sharing' application, the chosen technique was exploratory testing, which involves verifying and evaluating the product as it is being developed. As mentioned in the article by J. Itkonen and K. Rautiainen [13], some of the benefits of this testing method are:

- **Flexibility:** allowing for quick adaptation of tests and exploration of unplanned of less obvious scenarios
- **Efficiency:** prompt identification and reporting of issues enabling developers to address errors at an early stage of the development process
- **Obtaining an overall picture:** the reports contain valuable information about encountered problems and can be used to improve the application

Each step of the testing process is captured in a detailed report which contains important details such as: name of the tester, test duration, information about the application (version, tested functionalities, utilized scenarios etc.) and information about the obtained results.

In the following section, some of the reports that have been generated during testing phase will be presented.

In the following table 4.4 it is presented the report representing the testing of authentication feature, which was performed successfully, since the user's credentials were entered correctly.

Name	Iacob Victor Stefan
Session target	Authentication test
Duration estimation	10 minutes
Result	Success
Configuration	Start the application
Test	User authentication
Notes	The user was authenticated successfully

Table 4.4: Authentication testing report

In the following table 4.5 it is presented the report representing the testing of 'Add a car' feature, which was performed successfully, since the car was added to the list of 'MyVehicles' and it can also be found in the search results.

Name	Iacob Victor Stefan
Session target	'Add a car' feature test
Duration estimation	20 minutes
Result	Success
Configuration	Start the application
Test	Add a car
Notes	The car was added successfully.

Table 4.5: 'Add car' testing report

In the following table 4.6 it is presented the report representing the testing of 'Save car to favorites' feature, which was performed successfully, since the car was added to the list of 'MyFavorites'.

Name	Iacob Victor Stefan
Session target	'Save car to favorites' feature test
Duration estimation	20 minutes
Result	Success
Configuration	Start the application
Test	Save car to favorites
Notes	The car was saved into the favorites list successfully

Table 4.6: 'Save car to favorites' testing report

In the following table 4.7 it is presented the report representing the testing of 'Remove a car' feature, which was performed successfully, since the car was removed from the list of 'MyVehicles' and it cannot be found in the search results.

Name	Iacob Victor Stefan
Session target	'Remove a car' feature test
Duration estimation	15 minutes
Result	Success
Configuration	Start the application
Test	Remove a car
Notes	The car was removed successfully. It cannot be found in the search results

Table 4.7: 'Remove a car' testing report

In the following table 4.8 it is presented the report representing the testing of 'Search for a car' feature, which was performed successfully, since the results match with the ones predicted before starting of the testing phase.

Name	Iacob Victor Stefan
Session target	'Search for a car' feature test
Duration estimation	30 minutes
Result	Success
Configuration	Start the application
Test	Search for a car
Notes	Results match with the list predicted before

Table 4.8: 'Search for a car' testing report

In the following table 4.9 it is presented the report representing the testing of 'Edit profile' feature, which was performed successfully.

Name	Iacob Victor Stefan
Session target	'Edit profile' feature test
Duration estimation	10 minutes
Result	Success
Configuration	Start the application
Test	Edit profile
Notes	Profile data was updated. Profile image changed on the main page too after editing

Table 4.9: 'Edit profile' testing report

Chapter 5

Results and conclusions

This final chapter presents an overview of the achieved results in the development process of the bachelor thesis and highlights potential functionalities that can be further implemented in order to improve the application in the future.

5.1 Summary

In this thesis, it was presented a mobile application that is based on the concept of 'Peer-to-peer car sharing', which brings benefits from both environmental and economical points of view. While Romania can enjoy the reduction in the number of cars on the streets, users can also benefit from the functionalities of the application, gaining access to a platform that allows them to transform their personal vehicle from an active expense to a source of passive income. Additionally, renters can easily search for a car when they are in need of transportation, without having to pay large amounts of money into the maintenance of their personal vehicle. Moreover, they will find lower prices compared to traditional car rental centers and can choose from a wide range of vehicle, the one that meets all their requirements.

Looking back to the objectives set at the beginning of this thesis, it can be concluded that the application meets the expectations in terms of its functionalities and security provided for the users. Also, its scalability proves to be a significant benefit, since it enables the system to effectively handle the increased traffic, which can be a serious problem for this type of applications.

In comparison to other existing car rental/sharing applications in Romania, this one stands out, because of the fact that users can directly interact with each other without the need of a third instance.

Lastly, Ionic technology proves to be a key choice in the development process, as the application is available for both iOS and Android platforms making it accessible on over 99% of the existing smartphones. [8]

5.2 Further work

As time goes by and the number of users is increasing, the application must also be improved with new technology and innovative functionalities in order keep its audience. Due to the utilization of popular technologies in mobile application development (Ionic - Angular, Firebase) and intuitive code structure, potential updates should not require significant effort.

One way to increase the complexity of this application would be to implement a built-in payment method that should allow users to directly transfer funds for the rental of a car just by using the app.

Another interesting feature would be a recommendation algorithm that should display cars based on a specific user's preferences. This would simplify the search process, as the initial list would only consist of results that have higher chances of being chosen by that particular user.

Bibliography

[1] Sundararajan Arun. *The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism*. The MIT Press, 1st edition, 2016.

[2] Sophia Auer, Sophia Nagler, Somnath Mazumdar, and Raghava Rao Mukkamala. Towards blockchain-iot based shared mobility: Car-sharing and leasing as a case study. *Journal of Network and Computer Applications*, 200(1), 2022.

[3] Natalia Barbour and Yu Zhang. Individuals' willingness to rent their personal vehicle to others: An exploratory assessment of peer-to-peer carsharing. In *Transportation Research Interdisciplinary Perspectives*. ScienceDirect, 2020.

[4] Concept Apps Development SA. Citylink Romania, April 23, 2023. Screenshot of the CityLink application, Version 1.13.629.

[5] Jennifer Dill, Nathan McNeil, and Steven Howland. Peer-to-peer carsharing: Short-term effects on travel behavior in portland, or. In *Transportation Research and Education Center*. Portland State University, 2017.

[6] Dolan, S. (n.d.). How mobile users spend their time on their smartphones in 2023. `https://www.insiderintelligence.com/insights/mobile-users-smartphone-usage/`. Online; accessed 20 March 2023.

[7] Duncan Campbell. The secret to killer location queries? Meet the Geohash. `https://duncanacampbell.medium.com/demystifying-compound\-location-queries-in-firebase-740e88a3fa9a`. Online; accessed 20 May 2023.

[8] Thomas C. G and A. Jayanthila Devi. A study and overview of the mobile app development industry. *International Journal of Applied Engineering and Management*, 5(1), 2021.

[9] Gabriel Gircenko. Kotlin vs. Java: All-purpose Uses and Android Apps. `https://www.toptal.com/kotlin/kotlin-vs-java/`. Online; accessed 20 March 2023.

[10] Robert C. Hampshire and Srinath Sinha. A simulation study of peer-to-peer carsharing. In *2011 IEEE Forum on Integrated and Sustainable Transportation Systems*, page 159–163. IEEE, 2011.

[11] Turo Inc. Turo, April 23, 2023. Screenshot of the Turo application, Version 23.12.1.

[12] Ionic Documentation. Introduction to Ionic. `https://ionicframework.com/docs`. Online; accessed 19 March 2023.

[13] J. Itkonen and K. Rautiainen. Exploratory testing: a multiple case study. In *2005 International Symposium on Empirical Software Engineering, 2005.*, pages 10 pp.–, 2005.

[14] Barker Jacquie. *Beginning Java Objects: From Concepts to Code*. Apress, 1st edition, 2005.

[15] Mario Honrubia. Car Sharing: A New Sustainable Model for Mobility. `https://www.ennomotive.com/car-sharing-a-new-sustainable-model-for-mobility/`. Online; accessed 28 March 2023.

[16] Phillip Edwards. How to Develop Cloud-based Mobile and Web Applications with Firebase. `https://www.toptal.com/google/firebase-serverless-mobile-and-web-apps`. Online; accessed 24 March 2023.

[17] Ritesh Ranjan. The Mobile App Architecture Guide for 2023. `https://www.netsolutions.com/insights/mobile-app-architecture-guide/#hybrid-mobile-application-architecture`. Online; accessed 27 May 2023.

[18] Satinder Singh. Native vs Hybrid vs Cross Platform – What to Choose in 2023? `https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/`. Online; accessed 23 March 2023.

[19] Susan Shaheen and Adam Cohen. Carsharing and personal vehicle services: Worldwide market developments and emerging trends. In *International Journal of Sustainable Transportation*, page 5–34. Taylor & Francis, 2012.

[20] SIXT SE. SIXT rent. share. ride. plus., April 23, 2023. Screenshot of the SIXT rent. share. ride. plus. application, Version 9.96.1-11709.

[21] Spandita Hati. Swift Vs Objective C - Top Programming Languages Compared. `https://www.knowledgehut.com/blog/programming/swift-vs-objective-c/`. Online; accessed 22 March 2023.