

Urban Mobility Upgrade: Car Sharing Software Platform for Electric Vehicles

A DISSERTATION PRESENTED
BY
IOANNIS GKOURTZOUNIS
TO
THE UNIVERSITY OF NORTHAMPTON

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BSc (HONS) COMPUTING
IN THE SUBJECT OF
GENERAL COMPUTING

UNIVERSITY OF NORTHAMPTON
NORTHAMPTON, UK
APRIL 2018

© 2018 - *IOANNIS GKOURTZOUNIS*
ALL RIGHTS RESERVED.

Urban Mobility Upgrade: Car Sharing Software Platform for Electric Vehicles

ABSTRACT

Nowadays, more and more people live in the cities as they have access to essential resources that help them make their lives better. As cities get crowded, the need for transport leads to the demand for more journeys, which in turn creates a rapid growth of motorized vehicles. The majority of these vehicles use internal combustion engines and this has some serious negative impacts in our everyday lives like air pollution, congestion and traffic accidents.

Thus, a new personal transportation solution is needed. Our proposed solution consists of two parts. First, the description of a system that tries to bring two “green” ideas together: the use of Electric Vehicles (EVs) and in particular Battery Electric Vehicles (BEVs), as they produce zero harmful emissions and they have a high level of efficiency, in car sharing systems that offer mobility-on-demand. Such schemes reduce vehicle ownership and help mitigating the effects mentioned previously.

The second part of our solution is a software package for car sharing companies with EVs in their fleets. We analyzed use cases of administrators and customers of one-way car sharing schemes. Charging station data were collected related to the city of Bristol and we designed, implemented and tested: (i) a web platform for administrators, where the system can manage user requests and accept or deny them, ensuring that the highest number of users will be served, (ii) a mobile application

that lets users login, select stations and request vehicles.

In order to increase the effectiveness of the system, we developed two algorithms that assign available vehicles to users. We generated simulations of route requests with different criteria. The Long mode algorithm performed better with an average gain of 1.83% in efficiency rate in 120 requests per day when compared to the Short mode. We evaluated the system and showed that we offer a software solution that is reliable, secure, universal, maintainable and extensible and this concludes our proposed urban mobility upgrade.

Contents

1	INTRODUCTION	1
1.1	Background	1
1.2	Problem Statement	2
1.3	Aim & Objectives	3
1.4	Project Overview	4
2	LITERATURE REVIEW	5
2.1	Research Description	5
2.2	Urban Mobility Today	6
2.3	The Need for Electric Vehicles	10
2.4	More Challenges Ahead	16
2.5	Car Sharing Systems	20
2.6	Software Platforms	23
2.7	Review of Research	25
3	METHODOLOGY	29
3.1	Service Description	29
3.2	Use Cases	31
3.3	Data Collection	33
3.4	Requirements Analysis	34
3.5	Technology Resources	36

4	SYSTEM DESIGN	40
4.1	Architecture	40
4.2	Web Platform Design	42
4.3	Mobile Application Design	46
4.4	Database Design	48
5	IMPLEMENTATION	51
5.1	Development Phases	51
5.2	Web Platform Implementation	53
5.3	Mobile Application Implementation	56
5.4	The Platform Scheduler	59
5.5	Short Mode Decision Algorithm	60
5.6	Long Mode Decision Algorithm	63
6	TESTING	68
6.1	Test Strategy	68
6.2	Functional Testing	69
6.3	Non-Functional Testing	71
6.4	User Acceptance Testing	72
6.5	Decision Algorithms Results	74
6.6	Evaluation	77
6.7	Limitations	81
7	CONCLUSIONS	83
7.1	Summary	83
7.2	Future Work	86
8	REFERENCES	87
9	APPENDICES	93
9.1	Gantt Chart	93
9.2	Project Plan	95

9.3	Web Platform Class Diagrams	96
9.4	Mobile Application Class Diagrams	105
9.5	Summary of Test Cases	107
9.6	Route Generator Results	108
9.7	Virtual Demo	111
9.8	Presentation	111

Listing of figures

2.2.1	Urban mobility demand explodes, new challenges ahead.	7
2.2.2	Number of motorized vehicles in different countries from 1975.	8
2.3.1	Energy mixture in Germany 2010.	11
2.3.2	Basic parts of electric vehicles.	13
2.3.3	Energy efficiency benefits from electrification and connectivity.	14
2.4.1	Fast charger deployment in Europe.	17
2.4.2	Evolution of global electric vehicle stock.	19
2.5.1	State of car sharing worldwide.	23
3.1.1	System overview of the web platform and the mobile application.	30
3.2.1	Administrators use case UML diagram.	32
3.2.2	Users use case UML diagram, showing what users can do.	33
4.1.1	System architecture with web platform, database and mobile app.	41
4.2.1	Mockup design for the Dashboard page.	44
4.2.2	Mockup design for the Stations page.	45
4.3.1	Mockup designs for My Profile and User History screens.	47
4.3.2	Mockup designs showing the menu and the Open Map screen.	48
4.4.1	Entity relationship diagram of the database.	49
5.1.1	Phases of development of our software solution.	52
5.2.1	Overview of Spring MVC architecture for the web platform.	54
5.3.1	Interactions between Activity classes and system components.	57

5.5.1	Short mode decision algorithm flowchart.	61
5.6.1	Long mode decision algorithm flowchart (part I).	64
5.6.2	Long mode decision algorithm flowchart (part II).	65
6.5.1	Efficiency rate graphs when the system accepts 60 requests. . . .	75
6.5.2	Efficiency rate graphs when the system accepts 120 requests. . .	76
6.6.1	Server monitoring graphs during web platform operation.	78
6.6.2	Screenshot of the Dashboard page in the web platform.	79
6.6.3	Screenshot of the Manage Stations page in the web platform. . .	80
6.6.4	Screenshot of the Live Chats page in the web platform.	80
6.6.5	Screenshots of My Profile and User History screens.	81
6.6.6	Screenshots showing the menu and the Open Map screen. . . .	82
9.1.1	Project Gantt Chart.	94
9.3.1	Class diagram for Controllers package.	97
9.3.2	Class diagram for Entities package.	98
9.3.3	Class diagram for Repositories package.	99
9.3.4	Class diagram for Security and Utils packages.	100
9.3.5	Class diagram for Services package (part I).	101
9.3.6	Class diagram for Services package (part II).	102
9.3.7	Class diagram for Controllers, Entities and Repositories classes. .	103
9.4.1	Class diagram for Evsharingapp package.	105
9.4.2	Class diagram for Base and Entities packages.	106
9.4.3	Class diagram for Utils package.	106
9.6.1	Detailed efficiency rate results for 60 requests per day.	109
9.6.2	Detailed efficiency rate results for 120 requests per day.	110
9.7.1	Administrator logging in to the web platform.	112
9.7.2	Dashboard Panel page of the web platform.	113
9.7.3	User registration screen on the mobile application.	114
9.7.4	User login screen on the mobile application.	115
9.7.5	Profile screen on the mobile application.	116
9.7.6	Open Map screen on the mobile application.	117

9.7.7	Selecting request time on the mobile application.	118
9.7.8	Request vehicle screen on the mobile application.	119
9.7.9	Receiving request status on the mobile application.	120
9.7.10	User History screen on the mobile application.	121
9.7.11	The request saved as a route in the Dashboard Panel.	122
9.7.12	Updating route details in the web platform.	123
9.7.13	Managing Admin accounts in the web platform.	124
9.7.14	Adding a new Admin account in the web platform.	125
9.7.15	Managing user accounts in the web platform.	126
9.7.16	Managing stations details in the web platform.	127
9.7.17	Managing vehicles details in the web platform.	128
9.7.18	Simulation Panel page of the web platform.	129
9.7.19	Live Charts page of the web platform.	130
9.7.20	Selecting a specific station in Live Charts page.	131
9.7.21	Selecting a specific timeframe in Live Charts page.	132
9.7.22	The assigned vehicle to the user request in Live Charts page.	133
9.7.23	Navigation menu on the mobile application (part I).	134
9.7.24	Navigation menu on the mobile application (part II).	135
9.7.25	Navigation menu on the mobile application (part III).	136
9.7.26	Settings screen on the mobile application.	137
9.8.1	Project presentation.	138
9.8.2	Table of contents slide.	138
9.8.3	Introduction slide.	139
9.8.4	Problem domain slide.	139
9.8.5	Background slide (part I).	140
9.8.6	Background slide (part II).	140
9.8.7	Background slide (part III).	141
9.8.8	Background slide (part IV).	141
9.8.9	Aim and objectives slide (part I).	142
9.8.10	Aim and objectives slide (part II).	142
9.8.11	Service description slide (part I).	143

9.8.12	Service description slide (part II).	143
9.8.13	Service description slide (part III).	144
9.8.14	Implementation slide (part I).	144
9.8.15	Implementation slide (part II).	145
9.8.16	Implementation slide (part III).	145
9.8.17	Implementation slide (part IV).	146
9.8.18	Implementation slide (part V).	146
9.8.19	Decision algorithms slide (part I).	147
9.8.20	Decision algorithms slide (part II).	147
9.8.21	Testing slide (part I).	148
9.8.22	Testing slide (part II).	148
9.8.23	Testing slide (part III).	149
9.8.24	Testing slide (part IV).	149
9.8.25	Screenshots slide (part I).	150
9.8.26	Screenshots slide (part II).	150
9.8.27	Screenshots slide (part III).	151
9.8.28	Screenshots slide (part IV).	151
9.8.29	Screenshots slide (part V).	152
9.8.30	Summary slide (part I).	152
9.8.31	Summary, links slide (part II).	153
9.8.32	References slide.	153
9.8.33	End of presentation.	154

List of Tables

2.6.1	Car sharing mobile apps and their characteristics.	24
3.4.1	Requirements for administrators and the web platform.	35
3.4.2	Requirements for users and the mobile application.	36
3.5.1	Characteristics of different data exchange methods.	38
4.2.1	The pages of the web platform and their functionalities.	43
4.3.1	The screens of the mobile application and their functionalities	46
5.5.1	Short mode algorithm overview and steps.	62
5.6.1	Long mode algorithm overview and steps.	67
6.2.1	Overview of functional testing with test cases.	70
6.3.1	Overview of non-functional testing with test cases.	72
6.4.1	Specifications for alpha testing.	73
9.2.1	Tasks allocation for Phase I.	95
9.2.2	Tasks allocation for Phase II.	95
9.2.3	Tasks allocation for Phase III.	96
9.2.4	Tasks allocation for Phase IV.	96
9.5.1	Summary of test cases for the web platform.	107
9.5.2	Summary of test cases for the mobile application.	108

Acknowledgments

I would like to express the deepest appreciation to my advisor Emmanouil Rigas, who continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible.

1

Introduction

1.1 BACKGROUND

Today, cities are our homes. Every year more and more people move to new environments for many reasons. Seeking for a better job to advance our careers or for joining new communities, we see urban environments as a pool of opportunities for our lives. Transport is crucial for individual commuters and businesses, so a city should offer reliable means for our personal mobility. Urban development satisfies our needs but the increasing number of residents within a city introduces serious problems. More trips mean more vehicles with internal combustion engines (ICE) on the roads. Unfortunately, this has significant side effects to our environment, health and economy.

1.2 PROBLEM STATEMENT

Urban mobility systems have added the necessary means for people to access essential city resources. The need for transportation was the catalyst for the rapid growing in numbers of motorized vehicles in urban areas and this turned out to have some concerning negative consequences to our lives. Global carbon dioxide emissions are rising every year, while greenhouse gases make global warming tangible to many countries. More cars mean more traffic, more road accidents and more people dead or injured. Congestion harms the economy of the city when the transportation of goods slows down. Time, fuel and money are wasted for every alternative route that drivers have to take caused by traffic jams in peak hours.

New systems have emerged to satisfy our need for mobility-on-demand. In car sharing schemes, people rent and use vehicles only for their trips. Car sharing allows low income inhabitants to access vehicles without having to buy them. Reducing the vehicle ownership helps mitigating some of the negative effects mentioned. Such schemes attracted large organizations and now serve a large number of customers every day. But still, the dominant vehicle on the roads is based on internal combustion engines, and this keeps its effects tied to our journeys.

We need a new solution to upgrade our personal mobility. Not only should we adopt a new way of travelling across cities fast, effectively and without the downsides of an ICE vehicle, but we also need to embrace the sustainable development of the cities. One crucial ingredient for this upgrade is the management of vehicles and resources of the new system with a software infrastructure that should be easy, accessible, effortless and straightforward to install, manage and configure. It should also be universal, maintainable and extensible, so vehicles and transportation related companies will help commuters to eventually change their mobility habits towards a more sustainable future for the cities and its residents.

1.3 AIM & OBJECTIVES

Our solution consists two parts. First, the description of a new system that tries to bring two “green” ideas together: the use of Electric Vehicles (EVs) as the enabler of sustainable mobility, and car sharing systems. Those ideas can work together and they both seem promising enough. Second, the implementation of a software solution that will enable businesses to manage the use of EVs in one-way car sharing services. An online platform that will take into account the stations, the charging level and direction of each vehicle, the numbers of EVs at each station and the distances from vehicles to users. This online system will serve users who request a trip with EVs via their mobile phones.

The purpose of this dissertation is to offer a software solution to car sharing companies with EVs in their fleets. The system will be able to manage data from vehicles and users, and decide whether it will accept or deny a user request for a trip with the company’s vehicles. To accomplish that, the EVs have to be charged and the scheduling solution should take into account the charging level of each vehicle at any given time, along with their current position and destination. The underlying aim of the proposed system is: to manage user requests for hiring an EV to drive between two locations and accept or deny them, ensuring that the highest number of users will be served in a given time period (e.g. a day). We formulated a set of the most important objectives of our solution. First, the system should store all related data on a server, such as the stations, the vehicle characteristics, the routes and user accounts. Administrators will be able to alter data, like loading car characteristics or a map with station coordinates. Users will be able to create an account, login and make a request for a trip from an Android application in their mobile devices. The system should compute all the factors that affect the number of people that will be served in a given time period. It will handle user requests and decide if it will accept or deny them. Finally, the Android application will inform the user about the status of his request.

1.4 PROJECT OVERVIEW

The project consists of developing two applications: a web based application for administrators to manage stations, vehicle, routes and user accounts, and an Android app to let users create accounts and request a trip from one station to another. Monitoring and decision making about whether a request should be approved or not, will be performed by the web application.

The proposed software solution should be easy to be applied on a universal level, so a car sharing company with EVs can install the system and have it up and running, ready to accept customer requests. The most challenging part of this project is to implement the best algorithm in decision making, in order for the system to serve the highest number of users. For this purpose we have developed two algorithms, one that takes into account the future routes of the vehicles in the departure station and decide if these vehicles are free or not, and a more complicated algorithm that also considers the future routes of other vehicles and if they can substitute a current "locked/assigned" vehicle to make it free for use.

In the next chapter we conduct our research on urban mobility through the literature review. We present our findings that lead us to the first part of our solution, the convergence of EVs and car sharing systems. Then we focus on the methodology we followed and the design, implementation and testing of our software package, which is the second part of our solution. We evaluate the web and mobile applications and in the last chapter we summarize the most important ideas. This concludes a thorough exploration on both theoretical and practical aspects of an urban mobility upgrade.

2

Literature Review

2.1 RESEARCH DESCRIPTION

The research of this report is focused on three major subjects and how they influence urban mobility. First, we examine the increasing urban development and the need for transport. As cities get crowded, the need for transport is necessary as it adds value to the everyday life of the residents. This leads to the demand for more journeys, which in turn creates a rapid growth of motorized vehicles. We will concentrate on the effects of the increased motorization in the environment, the city traffic and its financial activities.

Next, we will explore ways for sustainable urban development. Cities need to provide a high quality of life to its residents that should also be eco-friendly. Renewable energy sources and electric-drive technology seem promising enough. Our main interest is the electric vehicle itself, the types of electric vehicles and

their impacts in urban mobility, the society and the environment. How can we improve electric vehicles and what challenges await? The third subject of our research is car sharing. Mobility systems that make use of sharing a vehicle between many drivers, have distinct characteristics. We review the benefits they offer and their convergence with electric vehicle technology.

2.2 URBAN MOBILITY TODAY

Nowadays, more and more people live in the cities as they have access to essential resources that help them make their lives better. Cities offer various events, entertainment, more jobs and commuting experiences to their residents (Milgram, 1970). The attraction of more residents lead to urban growth so cities and towns are able to support them. In fact, more than 3 billion people spend their lives in cities, which is about half of the world's population and this number constantly increases (Banister, 2005).

Increasing urban development translates into a greater number of passengers and transported goods within the cities (Rodrigue et al, 2006). Rodrigue et al also claim that travelling on regular basis is realized with faster transport modes, so journeys cover longer distances in the same time. Transport is essential for a healthy economy, so both individuals and businesses can enjoy the benefits of employment and economic growth (Banister, 2005). It is often the only way for residents to access jobs, health and education services that are vital to their well-being (Gwilliam, 2002).

The rapid increase of urban population inevitably boosts the total number of journeys residents make. Urban mobility systems have to satisfy this exploding demand and as Van Audenhove et al (2014) point out, new challenges are presented: "planet-related" like air pollution, noise and ecological footprint, "people-related" like traffic jam and its consequences, and "profit-related" like insufficient transport capacities, increasing motorization and limited parking spaces (Fig. 2.2.1). Indeed, the road transport industry is accountable for 74% of global carbon dioxide emissions, contributing to more greenhouse gases (GHG) and eventually to global

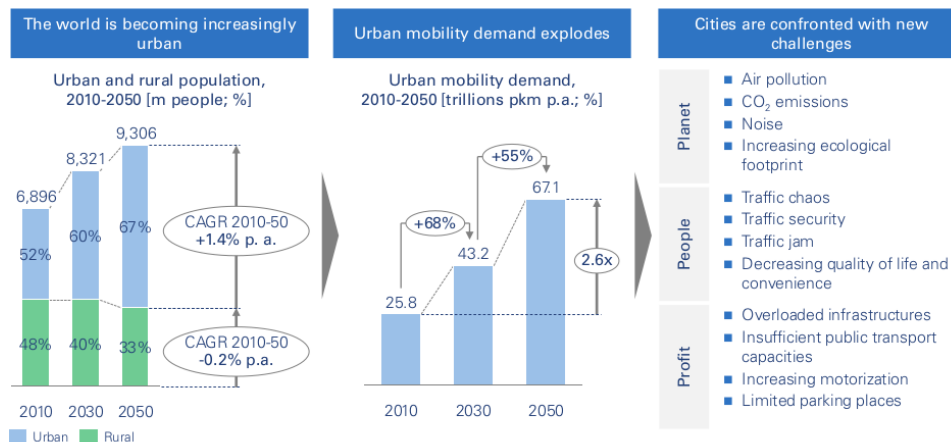


Figure 2.2.1: Urban mobility demand explodes, cities are confronted with new challenges (Van Audenhove et al, 2014).

warming (Rodrigue et al, 2006). This proportion is still rising in the developing countries as the number of vehicles is growing (Gwilliam, 2002). Additionally, particulate emissions are associated with health problems and transport noise affects the quality of life by disturbing or even traumatizing our ears.

If we look closely at traffic, transport safety and security, some troubling findings are waiting for us. Almost 500,000 people die and up to 15 million are hurt in urban road accidents every year in developing countries. Road accidents are the 9th cause of deaths worldwide and this figure is believed to climb to 6th by 2020 (Gwilliam, 2002). It appears that urban mobility systems were implemented fast enough to satisfy the demand, while there was no time for safety procedures to be developed.

Examining the transport system from a financial point of view, we will see that the wealth and well-being of the residents increasingly rely upon efficient road transportation. A road infrastructure that is blocked and choked leads to devastating consequences to the city economy and harms its low income inhabitants by slowing public transport. As Gwilliam (2002) puts it, a strategy for an efficient urban road system should be based on the mobility of people rather than the mobility of vehicles. Motorization rates are also climbing in advanced countries (Fig.

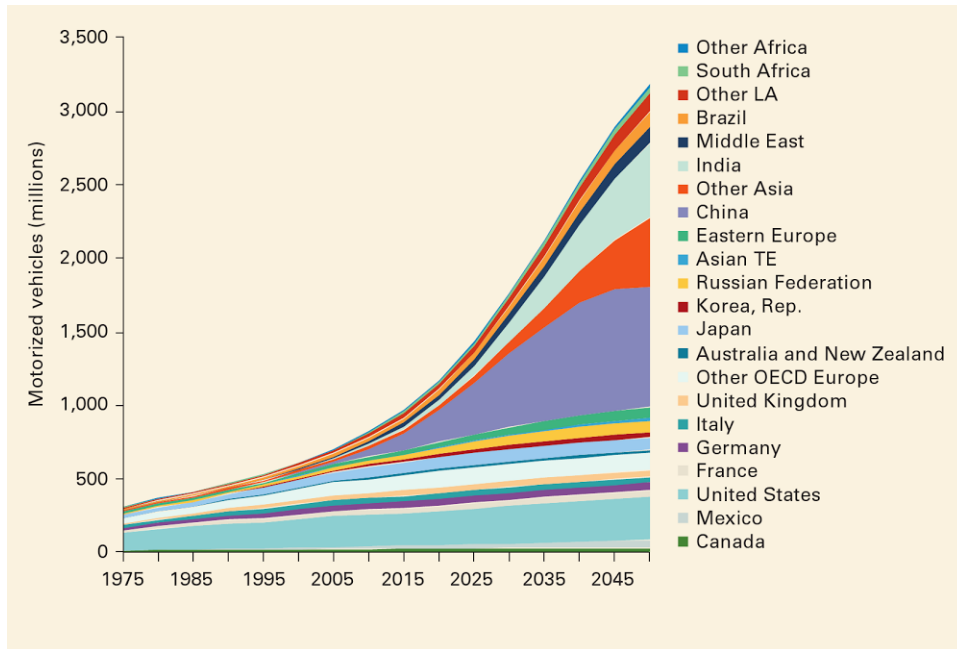


Figure 2.2.2: Number of motorized vehicles in different countries from 1975 (Suzuki et al, 2013).

2.2.2). High salaries fuel the vehicle ownership and modernization promotes a higher number of automobiles (Suzuki et al, 2013). This has some troubling implications such as more GHG emissions, which should be cut by at least 50% by 2050 for the climate to be stabilized (Sperling et al, 2009). Furthermore, Sperling et al also suggest that the more vehicles people own, the more quantity of oil we have to consume. It is expected that the demand will reach 120 million barrels of oil by 2030.

Unfortunately, more cars mean more journeys and more traffic in the cities. Road construction can not stay at the same level with the growing mobility demands. When the demand for transport is greater than the transport supply in a specific area, congestion occurs. In this situation, every vehicle hinders the mobility of others (Rodrigue et al, 2006). Congestion levels are so harsh in many large cities that they eventually harm the financial and social activities of its residents

(Sperling et al, 2009). Schrank et al (2007) in their mobility report, point out the main congestion problems: a lot of time, fuel and money are used just because people have to wait longer or take alternative routes. This leads to uncertain delivery times and missed meetings or relocations. Secondly, congestion affects the ones who make trips during the peak period. And thirdly, the congestion costs are increasing, estimated up to \$78 billion in 2005. The same results were confirmed in the mobility report of the same authors, five years later.

Moreover, congestion influences both personal trips and freight shipments in a negative way. Traffic congestion is disturbing to the people who experience it because of the time lost in traffic jams and the frustration due to extremely low driving speeds, causing psychological stress (Downs, 2004). Following the conclusion of Buchanan et al (2015), if we do not deal with the increasing motorization problems, either the residents will stop using vehicles in urban areas, or the enjoyment and safety of surroundings will disintegrate.

At this point, we will focus on the automobile itself as it is the most important element on the chain of transportation. Today most of the vehicles use internal combustion engines, as a large scale infrastructure is already built supporting this technology. Internal Combustion Engine or ICE vehicles use fossil fuels to generate mechanical energy but they are not effective, providing a low fuel conversion rate of about 30-35% (Mitchell et al, 2010). Additionally, ICE vehicles produce local air pollution and carbon dioxide emissions. Carbon dioxide concentrations that rise from burning fossil fuel, will lead to greater greenhouse effects and average global temperatures (McKay, 2008).

Another disadvantage of ICE vehicles is that fossil fuels are not renewable. There is only a finite quantity of oil and while we continue to consume it, the cost of finding, extracting and utilizing it, increases (Mitchell et al, 2010). Moreover, petroleum sources can be found only in a few geographic locations and this creates important security concerns for nations that import petroleum.

It is obvious that urban mobility today, serves all of us but with some crucial negative impacts in our everyday lives. More people in the cities have more transport needs and more vehicles. Increased motorization causes congestion, maximizing

the disadvantages of ICE vehicles. The GHG emissions in the year 2000 were about 34 billion tons of carbon dioxide equivalent per year (McKay, 2008). The consequences of climate change are global and they last for a long period of time, with irreversible changes waiting to occur (King, 2008). Congestion causes frustration, unreliable travel times and accidents. Time, fuel and money are wasted. The poor may encounter social and financial exclusion because of restricted personal mobility. The ICE vehicles are not even effective, providing a very low conversion rate. The demand in finite fossil fuels like oil, is rising like never before. How can we break the chains and create a better and more reliable personal mobility system?

2.3 THE NEED FOR ELECTRIC VEHICLES

Keeping in mind how urban mobility influences our lives and while recognizing the negative effects mentioned previously, we stand before a new global challenge. We need to accommodate motorization growth while conveying greater mobility and economic growth using sustainable approaches (King, 2008). But what does sustainable mean? Hopwood et al (2005) defines sustainable development as the outcome of the growing awareness of how environmental, social and financial problems affect our future.

In order to empower a sustainable mobility system, we need to take into account the transport infrastructure, the technological development of vehicles and the energy system that fuels them (Holden, 2007). Jenks et al (2005) point out that in order for the residents of a city to reach a high quality of life with ecological characteristics, they have to use renewable energy. This is an essential step to increase energy efficiency. The next logical step is the automobile itself. It is an integral component of the modern city life, so creating a new, effective automobile with a high conversion rate, is necessary for the sustainable development of cities (Kennedy et al, 2005).

Renewable energy sources seem to be the key in unlocking a new era of urban mobility. These are the energy sources that are not finite and will still be avail-

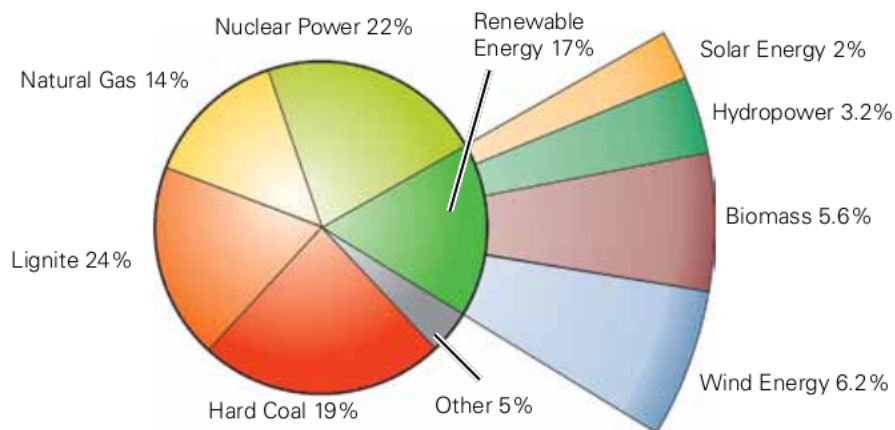


Figure 2.3.1: Energy mixture in Germany 2010 (Volkswagen Academy, 2013).

able even in continuous supply, like solar power, wind power, geothermics, hydropower and biomass (Herzog et al, 2001). The advantages they offer are that they do not pollute the atmosphere with GHG, they are freely available and their conversion and realization costs are decreasing over time (Droege, 2008). However, the difficulty in using these sources lies in collecting the actual energy, as the devices used for this purpose are not efficient enough (Mitchell et al, 2010).

Over the last years, more and more industries turn to renewable energy. In 2007, the share of renewable energy production globally has been estimated to be about 19%. However, 16% is produced by hydroelectric energy, while photovoltaic and wind remain some of the most promising sources (Lisserre et al, 2010). Volkswagen Academy (2013) in its study, revealed that renewable energy was the 17% of the total energy mixture in Germany in 2010 (Fig. 2.3.1). Moving towards that direction will allow cities to benefit from renewable energy sources and make them ready to welcome the next generation of automobiles.

The solution will be greatly enhanced by moving towards a new personal mobility system powered by electric-drive technology (Sperling et al, 2009). Despite the fact that 97% of all the vehicles today use combustion engines and burn

petroleum fuel, the next generation of automobiles will be propelled by electric motors. Electric-drive technologies include battery electric, hybrid electric, plug-in hybrids and fuel cell vehicles, which we will discuss later.

Now let us look at the main parts of an Electric Vehicle (EV). By studying the report *Basics of Electric Vehicles* from the Volkswagen Academy (2013), we recognize the following systems: a high voltage battery with a unit for regulation and charging, one or more electric motors and their cooling system, and a regenerative brake system. Reading further in this report, we acknowledge that the heart of the EV is the battery. It supplies voltage to the electric motors and they generate the mechanical energy needed to move the vehicle. By-wire systems transmit the electrical energy to the motors. In case of fuel cell vehicles, a fuel cell stack and hydrogen tanks are also parts of an EV (Fig. 2.3.2).

One of the most important advantages of EVs, is that electric motors are not only eco friendly with very low to none harmful emissions and noise, but they also provide good acceleration with a high level of efficiency (Volkswagen Academy, 2013). They effectively utilize more than 90% of the supplied energy, in contrast to the ICE vehicles where this proportion is no more than 37% (Sperling et al, 2009). Moreover, no energy is wasted when the car is at rest or when braking, instead the battery is recharged. If we use renewable energy sources to charge the battery, an EV can run emission-free in its lifecycle.

On the other hand, there are some important disadvantages as well. EVs offer a limited mobility range due to their battery size, charging can take up to 10 hours, and the charging infrastructure is sparse (Volkswagen Academy, 2013). This can create "range anxiety" and the driver will be influenced in a negative way, keeping in mind that he has to charge the vehicle before reaching its maximum range. So, fuel distribution systems is one of the biggest challenges for incorporating EVs in our urban mobility lifestyle (Sperling et al, 2009).

By upgrading our EVs with the necessary connectivity features, we can add substantially to its positive impacts. We can combine GPS and wireless communications to provide real time navigation, traffic information, remote diagnostics and

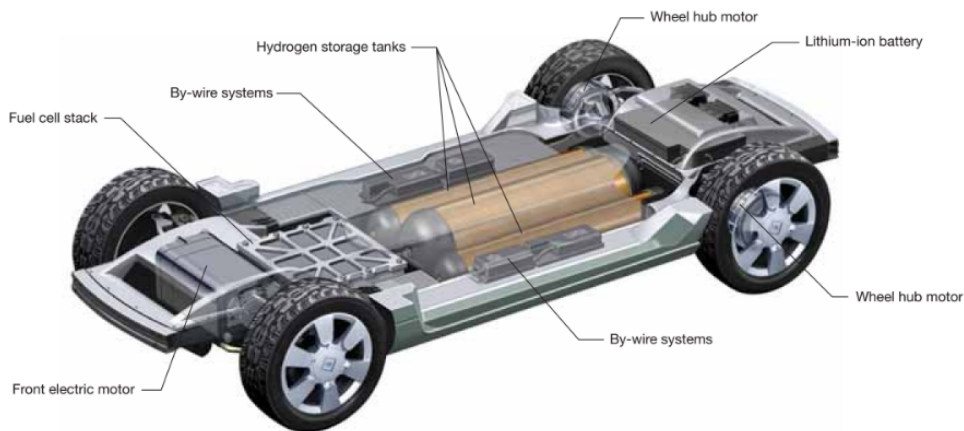


Figure 2.3.2: Basic parts of electric vehicles (Mitchell et al, 2010).

automatic crash notifications to authorities (Mitchell et al, 2010). The combination of EVs and connectivity improves traffic flow, reduces the number of crashes and it also reduces air pollution and energy consumption. With vehicle-to-vehicle communications, each car can broadcast its position and velocity, resulting in greater knowledge of the external environment while driving. Timely traffic information will make the trips of drivers shorter, predictable and less stressful (Mitchell et al, 2010). If the vehicle can operate autonomously, the "driver" can relax, work or do some social networking. So, apart from the obvious advantages of a lighter vehicle, adding connectivity to EVs offers a smoother travel with improved knowledge of road conditions and parking availability, and better driver performance with a more efficient vehicle operation (Fig. 2.3.3).

From the above, we reach to the same conclusion that Helmers et al (2012) state at the end of their article: the electric car is an essential ingredient for a more sustainable mobility future, although it has some drawbacks. Electrification provides higher efficiency, lower energy costs for personal mobility and enables energy diversity. Mitchell et al (2010) characterize EVs as clean, compact vehicles with a pleasurable driving experience where their cost becomes high only when they are required to provide a range of 100 miles or more at high speeds. This is not a disadvantage at all, as trips in the cities tend to be shorter and in low speeds.

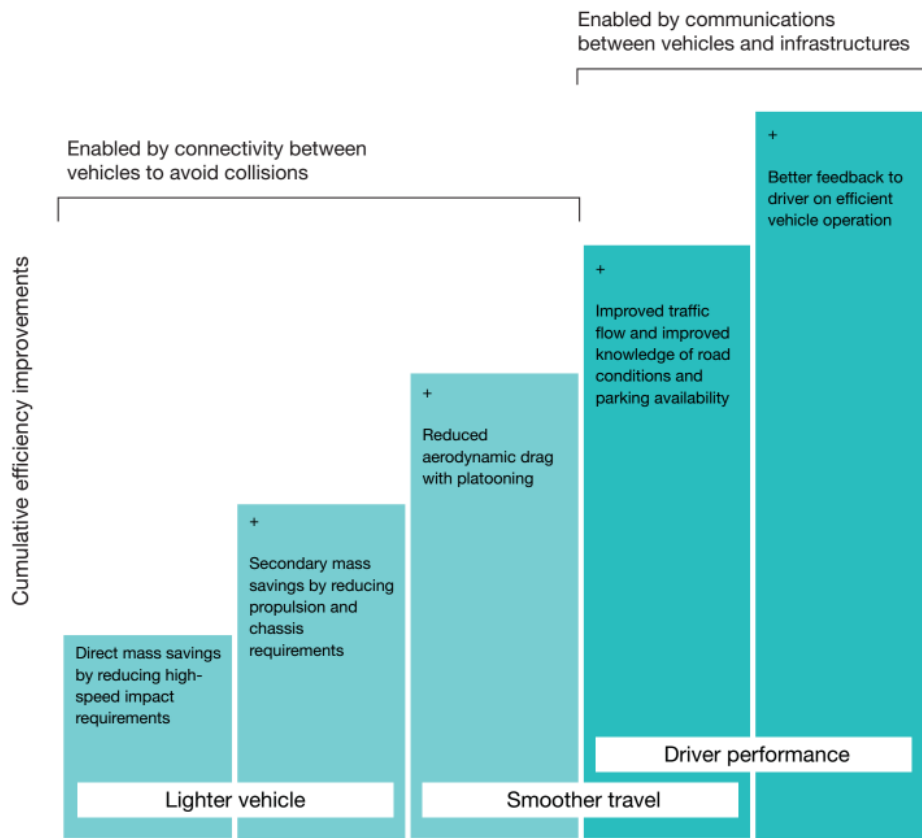


Figure 2.3.3: Energy efficiency benefits from electrification and connectivity (Mitchell et al, 2010).

Electric-drive technology includes many types of vehicles. We will now present them along with their basic features as they are stated in the report *Electric Vehicles in Europe* by the European Environment Agency (2016). Battery Electric Vehicles (BEVs) are powered only by an electric motor that uses electricity stored in a battery. The battery should be charged regularly by plugging in to a charging point. BEVs provide the highest energy efficiency with zero emissions, but they have a limited driving range and they need a long time to recharge.

Hybrid Electric Vehicles (HEVs) use both an internal combustion engine and an electric motor that assists the engine during driving. In those vehicles the battery can not be charged from the grid, it is charged only when the car is coasting, or during regenerative braking. HEVs have lower fuel consumption and emissions than ICE vehicles but they still rely on fossil fuels. Plugin Hybrid Electric Vehicles (PHEVs) share the same characteristics as HEVs with the difference that the battery can be charged from the grid, so they provide a longer driving range.

Fuel Cell Electric Vehicles (FCEVs) are powered only by electricity generated by fuel cells that combine hydrogen from tanks and oxygen from the air. FCEVs have significantly longer driving range and are faster in refuelling. However, fuel cells technology is in an early stage of development and FCEVs commercial availability is limited.

In this report, we consider the BEVs that use energy from renewable sources as the next step to our urban mobility upgrade. They have excellent room for progress, we can design lithium-ion batteries with ranges up to 240km and they present a satisfactory acceleration performance (Burke, 2007). Mitchell et al (2010) and McKinsey (2014) show that those vehicles can be recharged in less than 3 hours (fast charging) with a 240V outlet or in about 8 hours (slow charging) with a standard 110V outlet. Mitchell et al also add that with zero emissions, BEVs are the most affordable solution for limited range urban applications.

2.4 MORE CHALLENGES AHEAD

We will now take a closer look at BEVs and their challenges. In 2015, BEVs reduced the GHG emissions relative to ICE vehicles but they pose important health impacts as the manufacturing process for BEVs creates damaging environmental issues (Brennan et al, 2016). Nevertheless, the European Environment Agency (2016) disagrees and clearly states that during the vehicle's lifetime, the zero emissions cancel out the the environmental effects of the production and end-of-life phases. Furthermore, BEVs can use electricity exclusively generated from renewable sources, adding up to their environmental benefits.

Brennan et al (2016) in their assessment report argue that the total cost of ownership for a BEV is much greater than the cost of an ICE vehicle. In 2015, BEVs were more expensive to manufacture mainly because of the battery production cost. The development of battery technology is crucial and the price of the battery is the main concern for the people who are interested in buying a BEV. It has a large cost in the lifecycle of the vehicle as it has to be replaced after 10 years since the original purchase. A small BEV is still 30% more expensive than an equivalent ICE vehicle. Consequently, as both Millard-Ball et al (2005) and McKinsey (2014) point out in their reports, BEVs have been used for research pilots or for programs where funding was available explicitly for electric vehicles. King (2007) acknowledges that the main challenge for BEVs is to create high power and long life batteries with low cost. Good performance EVs with low cost is essential for leveraging the initial investment and attracting new businesses (Chan, 2002).

There are three ways that a vehicle can be charged: swapping, wireless and wired (McKinsey 2014). Battery swapping was incorporated in the Tesla Model S vehicle, but lost its appeal because it was too costly. Induction charging, where electricity is transferred wirelessly, allowed only for lower transfer rates and its implementation was also quite expensive (Mitchell et al, 2010). The third and most used option is wired or plug-in charging where the vehicle is connected to the grid through a charging point (McKinsey 2014).

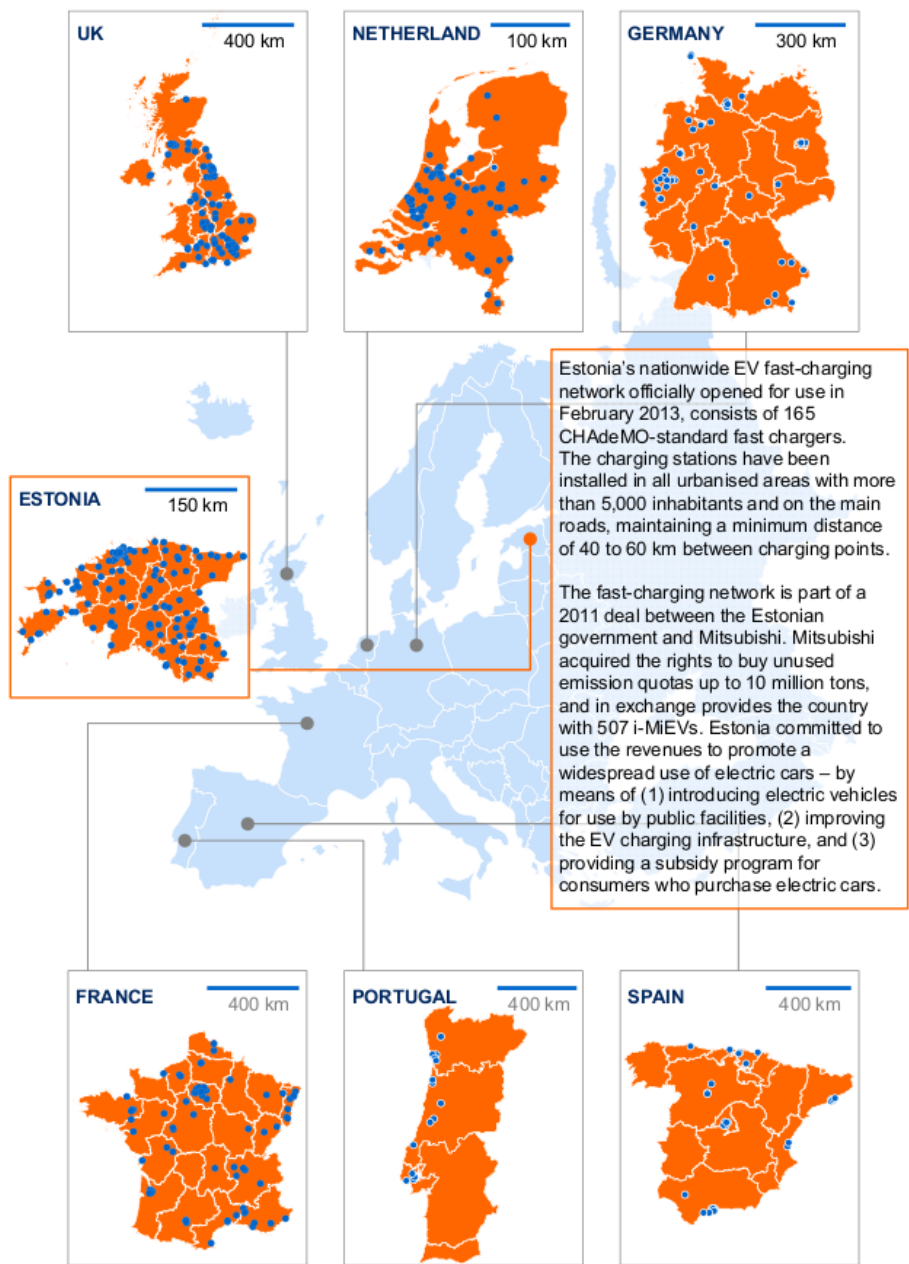


Figure 2.4.1: Fast charger deployment in Europe (McKinsey, 2014).

The success of BEVs depends heavily on the charging infrastructure. The allocation of charging points on the roads should eliminate the "range anxiety" (Mitchell et al, 2010). Burns et al (2002) in their article *Vehicle of Change*, uncover the so called "chicken and egg problem": large numbers of EVs require enough fuel availability, but at the same time the necessary charging infrastructure can not be built if there is not a significant number of vehicles on the roads. A good charging infrastructure is essential for the commercialization of the BEVs in today's market.

To confront this challenge, the cost of charging stations are decreasing and new businesses are entering the field of EV charging (McKinsey 2014). Furthermore, fast charging stations can provide enough energy for a 100 mile trip in just a 10 minute charge (Botsford et al, 2009). As a result, the expansion of charging infrastructure in 2015 was about 71%, comparable to the growth of the number of EVs globally (Global EV Outlook, 2016). In Europe that year, Netherlands had a network of more than 23,000 charging points, while Germany, France and the UK followed with significant numbers (European Environment Agency, 2016). From 2011, Estonia was committed to promote the use of electric cars and its fast charging network and is still growing rapidly (Fig. 2.4.1). However, more fast-charging points with easy access need to be implemented, along with smart grid devices (Garcia-Valle et al, 2013).

Apart from the infrastructure, the process of EV charging heavily depends on the electrical grid. The grid is the platform for delivering electricity to millions of people in a second's notice (Schewe, 2007). Here lies the next big challenge, how the grid will react in the potential increase in demand during peak hours (McKinsey, 2014). Mitchell et al (2010) inform us about the possible issues: the grid capacity, electricity trading, load balancing, and the quality of electric supply. A possible solution is coordinated charging, where flattening out peak power can eliminate power losses and voltage alterations. As Clement-Nyns et al (2010) state, the coordination can be done with smart metering devices and by sending signals to the individual vehicles. Smart grids can use many distributed generators in order to satisfy energy demand and avoid excessive load peaks (Lisserre et al, 2010).

Mitchell et al (2010) propose a few more ways to tackle the problem of load

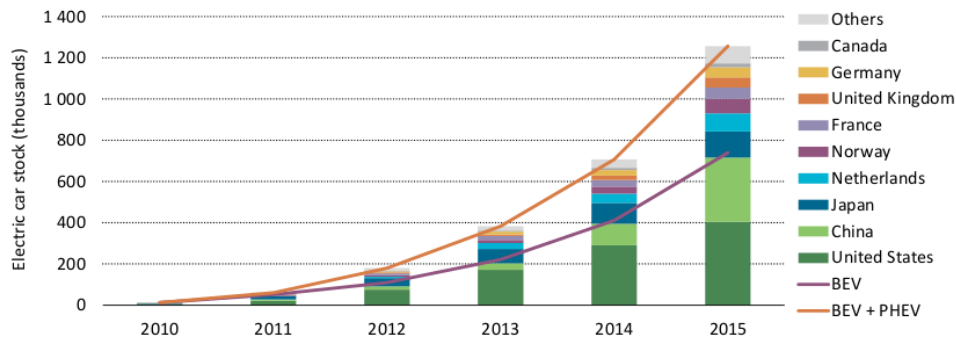


Figure 2.4.2: Evolution of global electric vehicle stock (Global EV Outlook, 2016).

peaks and enhance the effectiveness of the grid. One is to use BEVs as a “free battery storage” along with coordinated charging. Another way is to introduce dynamic electricity pricing to keep supply and demand in equilibrium. Prices can increase when the load on the grid is high, and they can decrease when the demand is low. In the last years active efforts are being made to develop and implement such smart grids.

So, what actions are necessary to deal with these challenges? First, we need to develop consistent incentives to motivate people and organizations. Second, the time is now to advance a broad portfolio of energy-efficient, low-carbon technologies (Sperling et al, 2009). Turrentine et al (2010) in their report are more specific by proposing the following recommendations: increase consumer awareness for EVs and reduce their costs, simplify and prioritize home charging installations, optimize placement of charging infrastructure, ensure the reliability and safety of the grid and support fleet purchases of EVs. In order for a viable EV market to emerge, OEMs, suppliers, power companies and governments need to work together (Dinger et al, 2010).

Indeed, we can see that urban transport is now on the EU policy agenda with significant funding for research and technology innovation in order to support cities moving towards a sustainable mobility system (Gaggi et al, 2013). Europe initiatives like low carbon dioxide emissions, reduction in taxes and ownership costs,

financial support to the EV industry, and local government actions (e.g. free parking places, free charging, reserved lanes) are just some of the measures that are promoting EV use today (European Environment Agency, 2016). And those measures are giving results. Policymakers and engineers focused in local air pollution with great success in certain nations (Sperling et al, 2009). The global threshold of 1 million EVs on the roads was surpassed in 2015 and reached 1.26 million at the end of that year (Fig. 2.4.2). New registrations of EVs have grown in numbers by 70% between 2014 and 2015, reaching over 550,000 sold vehicles worldwide in 2015 (Global EV Outlook 2016).

2.5 CAR SHARING SYSTEMS

Nowadays, most consumers care about the sustainability of their mode of travel and are willing to adapt new mobility systems such as car sharing and bike sharing. Van Audenhove et al (2014) mention in their study that car sharing has emerged from a community-based collaboration, to a big business that attracts major vehicle manufacturers and younger customers.

But how does car sharing fit to our personal mobility in the cities? Usually an organization purchases a fleet of cars and makes them available to the residents who access them on an as-needed basis for their mobility demands. In these systems the vehicle usage is booked in advance, customers access the cars themselves and rentals are for short periods of time (Millard-Ball et al, 2005). They differ from ride sharing or carpooling as their provided access is about short trips and the charge is valid only for the duration of the trip (Katzev, 2003).

Three basic car sharing types are examined by Laarabi et al (2016). In two-way systems the vehicles can be picked up from any of the preconfigured stations, but they have to be dropped off at their initial station. In one-way systems (e.g. Zipcar, Modo) the vehicles can be picked up or dropped off at any of the stations. Free floating car sharing systems (e.g. Car2go, DriveNow, Enjoy) allow for maximum flexibility as there are no stations and the customer can use any public parking within the area of operation. In such mobility-on-demand schemes, we can

apply algorithms to increase the number of customers being served, thus making those schemes even more effective (Rigas et al, 2015).

In this report we will concentrate in one-way car sharing service. In this paradigm the customer looks for the closest station and when an available vehicle is found, he proceeds on booking the vehicle for a trip. He picks up the vehicle, makes his short trip, books the closest station to his destination and finally drops off the vehicle (Laarabi et al, 2016). This service can also be used with EVs. If we use BEVs, we just have to make sure that the stations can also charge the vehicles and that the vehicles are available only if their charge level is above a certain threshold.

Adopting car sharing systems in urban mobility has many positive impacts. It is referred as the missing link of the alternatives to the private vehicle, like taxis, cycling and walking (Millard-Ball et al, 2005). Vairani (2009) shows in his report one of the most important benefits: the costs for vehicle access are divided among a group of people, so the low income inhabitants do not need to pay large upfront costs. As Millard-Ball et al (2005) add, costs that depend on driving time is a strong financial incentive to drive less. The need for privately owned vehicles along with their negative consequences like GHG emissions, traffic, congestion, parking, etc. are reduced, which essentially means more efficient land use.

Car sharing offers great mobility to people by allowing them to travel without owning a car and it is most cost effective for intermediate length trips (Millard-Ball et al, 2005). Consumers do not want to buy or own things, they prefer to pay for temporary accessing them (Bardhi et al, 2012). A reduction in vehicle ownership leads to a 28% - 45% reduction in vehicle miles travelled and 19% - 54% lower GHG emissions for the average driver (Shaheen et al, 2007). Fellows et al (2000) show that those benefits, along with increased speeds and fuel savings, are comparable to major road schemes with just a fraction of the implementation costs. Studies suggest that each vehicle from car sharing removes 6 to 23 cars from US roads and 4 to 10 cars from European roads (Vairani, 2009). As we saw, relying on sharing cars rather than owning them, will be the greater environmental benefit of car sharing (Katzev, 2003).

Let us now look at the main challenges of car sharing programs. One of the

most serious problems of one-way car sharing is that the distribution of the vehicles should allow availability to customers at all times. Instead, bottlenecks in vehicles supply will create areas of low mobility demand with a large number of vehicles and areas without enough vehicles left to satisfy high mobility demands (Weikl et al, 2013). The strategies that can solve this problem are user-based and operator-based according to Weikl et al (2013). In user-based relocation strategies, customers should be given incentives or lower rates to move vehicles to "cold spot" areas when needed. Mitchell et al (2010) agree that dynamic pricing can make spatial distribution of vehicles meet the mobility demand. In operator-based strategies, the staff can relocate the vehicles or buffer depots can release standby vehicles when the demand is high (Weikl et al, 2013).

Another challenge is that the software behind car sharing systems pose threats of the location privacy of the drivers as some implementations involve pervasive tracking (Popa et al, 2009). Fortunately, there are some solutions on this matter, like Vpriv, which is a protocol for protecting the location privacy of the drivers while supporting applications that deal with paths travelled by individual cars.

Car sharing started gaining large ground in Switzerland and Germany, where the first programs served their residents in the late 1980s (Millard-Ball et al, 2005). Since then, more and more people and businesses participate in these programs. In 2007, car sharing was a viable solution in 600 cities around the world with 348,000 drivers sharing roughly 11,700 vehicles (Shaheen et al, 2007). As Shaheen et al inform us, some of the largest organizations became multinational operators as seen in Fig. 2.5.1, like Zipcar, Greenwheels, Cambio Car and CityCarClub. The biggest car sharing company now is Zipcar which by the end of 2011 had more than 650,000 members and 8,900 cars in urban areas worldwide (Bardhi et al, 2012).

The convergence of EV technology and car sharing is also taking place. Some of the companies that include EVs in their fleets are Car2Go, DriveNow, Flinkster and Autolib (McKinsey, 2014). Both technologies are growing and influence one another with great positive impacts to urban mobility. More people get familiar with EVs and that will lead to a bigger pool of potential buyers. Taking into account

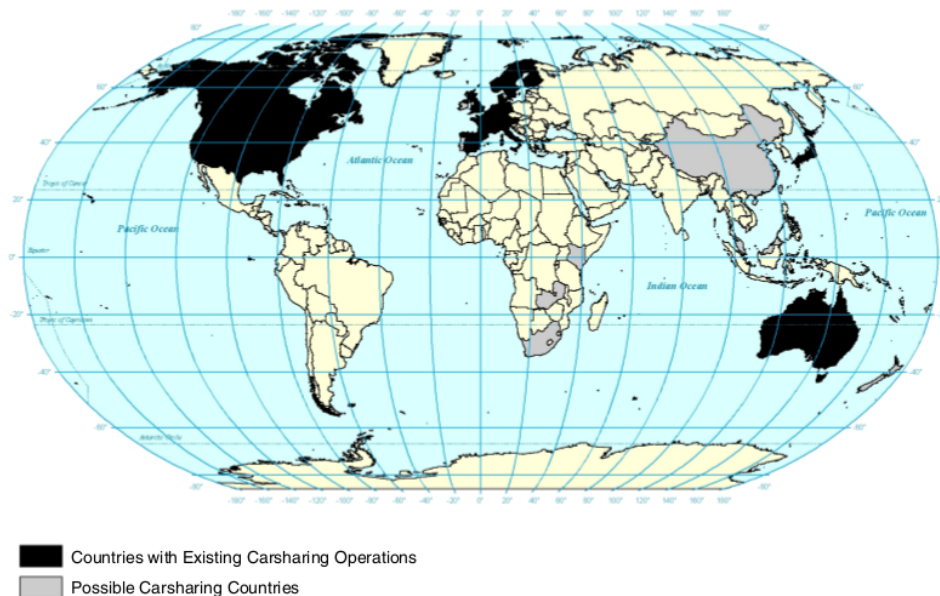


Figure 2.5.1: State of car sharing worldwide (Shaheen et al, 2007).

the advantages of BEVs, we get the best of the EV world, and utilizing them in one-way car sharing schemes allows us to benefit from both worlds. Car sharing can multiply the positive impacts of EVs to the society as a whole. It can also be the solution to "range anxiety" of drivers when thinking of purchasing an EV, allowing them to use BEVs for distances that they feel comfortable (McKinsey, 2014). So, renewable energy, BEVs and car sharing can be the next big thing in our urban mobility upgrade.

2.6 SOFTWARE PLATFORMS

Many car sharing companies include EVs in their fleets for their customers' trips. Although they have EVs available, their mobile applications are general and not tailored around EVs and their special needs, like charging stations and individual vehicle charging level. This is a major drawback, as the true benefits of EVs and car sharing schemes can be revealed only if the software solution is tailored to EVs and their special characteristics.

Our research for companies that offer car sharing services with EVs ended up with just a few car sharing schemes that offer mobile applications to their users. From Paris, London and Indianapolis, to Boston, Michigan, Vilnius and Sofia, companies are offering their services through their customized applications for their customers' mobile phones as seen in Table 2.6.1.

CAR SHARING APPS	CHARACTERISTICS
Autolib	A full EV service with Bluecars in Paris
Bluecity	London's first full EV scheme
Blueindy	EV car sharing scheme in Indianapolis
Bonzer from Instacarshare	Car sharing with mini EVs in Boston
MDrive from Rent Centric	With three Ford Focus EVs in Michigan
SPARK	EV car sharing in Vilnius and Sofia

Table 2.6.1: Car sharing mobile apps and their characteristics.

As we can see, each company has developed its own software platform and a mobile application that manages requests from the users. Their software solutions are applicable to their fleets and their specific business objectives. Our purpose is to expand that and offer a universal solution which can be easily customized for any car sharing company.

A universal solution means that an online platform for managing administrators, users, stations, vehicles and routes will be ready to be delivered to a company, along with a working mobile application that will let the users request a vehicle for a trip. Instead of having each company develop their own software, a universal solution could speed up the establishment and operation of new companies interested in "one-way car sharing with EVs" schemes. Software development costs could be reduced and extensibility would ensure that those companies could expand their operations seamlessly. Another advantage of having many different car sharing schemes use the same platform is the opportunities for data analytics and improvement of algorithms used to assign vehicles to users, when two or more

companies analyze and compare their data together.

After the review of research, in the following chapter we describe the methodologies used to elicitate the requirements for this project. Next, we design, implement and test our software solution, which consists of a web based application as the online management platform and a mobile application for Android devices.

2.7 REVIEW OF RESEARCH

Our research starts with urban mobility, where the article *The Experience of Living in Cities* (Milgram, 1970) presents the characteristic qualities of city life. Rodrigue et al in their book *The Geography of Transport Systems* (2006), examine how transportation helps in the development of cities, but when the number of vehicles increases, there are serious urban and environmental impacts. The *Cities on the move: a World Bank urban transport strategy review* (Gwilliam, 2002) focuses on the relationship between urban transport and the city development. It shows that transport and increased motorization affects the financial life of commuters and hurts their health not only by the rising GHG, but also with road accidents.

The *Urban Mobility Reports* by Schrank et al in 2007 and 2012 for the Texas Transportation Institute, reveal various congestion related problems and the negative effects to the residents of the cities. Congestion is the subject thoroughly examined in the book *Still Stuck in Traffic: Coping with peak-hour traffic congestion* (Downs, 2004), with its causes and consequences. *Traffic in Towns: A study of the long term problems of traffic in urban areas*, a popular study by Buchanan in 2015, acknowledges the long term problems of traffic in urban areas and it proposes policies for reducing congestion in favor of the quality of life of the commuters.

The links between transport and sustainable urban development are identified and analyzed in *Unsustainable Transport: City transport in the new century* (Banister, 2005). In their study *Transforming Cities with Transit: Transit and land-use integration for sustainable urban development*, Suzuki et al (2013) explore the increased motorization in urban areas and how it transformed our cities. They recommend policies to overcome current obstacles and promote sustainable urban develop-

ment. Gaggi et al, (2013) also talk about the EU policy agenda and sustainable transport planning.

The idea of sustainable development is defined and further explored in the article *Sustainable Development: Mapping Different Approaches* by Hopwood et al (2005). Kennedy et al (2005) recommend the *Four Pillars of Sustainable Urban Transportation*, which are: effective governance of land use and transportation; fair, efficient, stable funding; strategic infrastructure investments; and attention to neighbourhood design. Policies for achieving sustainable mobility are evaluated in *Achieving Sustainable Mobility: Everyday and leisure-time travel in the EU*, where Holden (2007) also argues that leisure-time travel, previously largely neglected, should be included in any sustainable mobility policies. *Future Forms and Design for Sustainable Cities* by Jenks et al (2005) concentrate on the planning and design of cities, when Sperling et al (2009) tell us why and how we need to transform transportation now more than ever in their book *Two Billion Cars: Driving toward sustainability*.

Renewable energy technologies are presented in the article *Renewable Energy: A Viable Choice* by Herzog et al (2001). Droege in his book *Urban Energy Transition: From Fossil Fuels to Renewable Power* in 2008, explains how we can make the change and shift from cities dominated by the fossil-fuel systems of the industrial age, to a renewable-energy based urban development framework. Preparing for that change on a personal and international scale is also the main subject of the case study *Sustainable Energy: Without the hot air* (McKay, 2008). The report *The King Review of Low-Carbon Cars Part I* in 2007 talks about the potential of the reduction of carbon dioxide emissions, where Part II in 2008 presents policy recommendations in regards to renewable energy.

Van Audenhove et al (2014) presented an in-depth exploration about the future of urban mobility with *The Future of Urban Mobility 2.0: Imperatives to Shape Extended Mobility Ecosystems of Tomorrow*. The study highlights what is holding cities back and identifies three strategic directions for cities to better shape their mobility infrastructures: countries with high motorization rates need to fundamentally redesign their mobility systems, increase the overall attractiveness of public trans-

port, and establish a sustainable mobility core that can satisfy short term demand at a reasonable cost.

Sustainable urban development inevitably leads us to a new era of vehicles. In their article *Vehicle of Change* in Scientific America, Burns et al (2002) presented EVs and argued that the transition to FCEVs could transform energy infrastructures while helping the environment. Seven years later, Vairani in his report introduced *bitCar* (2009), a design concept for a collapsible stackable city car powered by electric motors. The book *Reinventing the Automobile* by Mitchell et al (2010), provides a great insight of a long-overdue vision for a new automobile era. The Automobile is reimaged and at the heart of a more convenient and sustainable urban mobility lies the EV.

EVs, their characteristics and types are described in the study *Basics of Electric Vehicles* by Volkswagen Academy (2013). Helmers et al (2012) focus on the environmental impacts of BEVs with their article *Electric Cars: Technical Characteristics and Environmental Impacts*, while the key technologies and commercialization of HEVs are discussed in *The State Of The Art Of Electric And Hybrid Vehicles* (Chan, 2002). Burke (2007) explores the potential of batteries and ultracapacitors and how PHEVs can be designed with effective all-electric ranges. Brennan et al (2016) in their assessment report *Battery Electric Vehicles vs. Internal Combustion Engine Vehicles* showed that the total cost of ownership of BEVs is still a burden for consumers.

Dinger et al (2010) provide an in-depth report titled *Batteries for electric cars: Challenges, opportunities, and the outlook to 2020* that analyzes the impact new batteries will have on the emerging EV market, the technological barriers for lithium-ion batteries, and the cost of batteries as the technology develops. Fast charging and slow charging are presented with their strengths and weaknesses relative to battery charging infrastructure, by Botsford et al in their article *Fast charging vs. slow charging: Pros and cons for the new age of electric vehicles* (2009).

Charging EVs cause extra electrical loads that impact the distribution grid. The power losses and voltage deviations are analyzed in the article *The Impact of Charging Plug-In Hybrid Electric Vehicles on a Residential Distribution Grid* (Clement-Nyns

et al, 2010). Software tools, along with dynamic operation of electricity grids, are proposed in the book *Electric Vehicle Integration into Modern Power Networks* by Garcia-Valle et al (2013). The grid is explored in *The Grid: A Journey Through The Heart Of Our Electrified World* (Schewe, 2007), and its transformation into a smart power grid that exploits renewable energy is discussed by Liserre et al with their article *Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics* (2010).

EVs are increasing rapidly mainly because of increased consumer awareness, cost reduction and home charging installations prioritization, as Turrentine et al (2010) show in their report. McKinsey in *Electric Vehicles In Europe: Gearing up for a new phase?* (2014), also show that OEMs, suppliers and governments should work together to tackle implications related to the grid and the charging stations in order for new business models and market opportunities to arise with the wide adoption of EVs. *Electric Vehicles in Europe*, a report by the European Environment Agency (2016), presents the results of such policies: the number of charging stations is constantly increasing and this boosts the number of EV stock globally as shown in *Beyond One Million Electric Cars* by Global EV Outlook (2016).

Our research concludes with car sharing, described in the articles *Car Sharing: A New Approach to Urban Transportation Problems* (Katzev, 2003) and *Access-Based Consumption: The Case of Car Sharing* (Bardhi et al, 2012). The basic schemes of car sharing are discussed, with their environmental benefits and their greater adoption from companies in many countries.

Software frameworks, relocation strategy, and privacy protecting algorithms have been developed to ensure proper operation of such schemes, are discussed by Laarabi et al (2016), Weikl et al (2013) and Popa et al (2009) respectively. Rigas et al in their article *Algorithms for Electric Vehicle Scheduling in Mobility-on-Demand Schemes* (2015) showed that in such mobility-on-demand schemes, we can apply algorithms to increase the number of customers being served. Car sharing is evaluated and its environmental, economic and social impacts are presented in *Car-sharing: Where and how it succeeds* Millard-Ball et al (2005), while Shaheen et al (2007) forecast continued growth, particularly among new and emerging markets.

3

Methodology

3.1 SERVICE DESCRIPTION

The objects that need to be saved and managed in a car sharing software solution are the stations, vehicles and routes. Routes are essentially a combination of a station, a vehicle and a user. Requests are used to create routes and administrators need to be managed and added. Requests, user accounts and administrator accounts should also be saved. Our proposed solution is a system that consists of an online web application (platform) and a mobile application. Its purpose is to serve two kinds of audiences, the administrators and the users, and manage their data.

The online web platform is the backend interface to manage the system. It can be installed on a server and allows the administrators to manage users, stations, vehicles, routes and accept or deny user requests. Administrators can login, save,

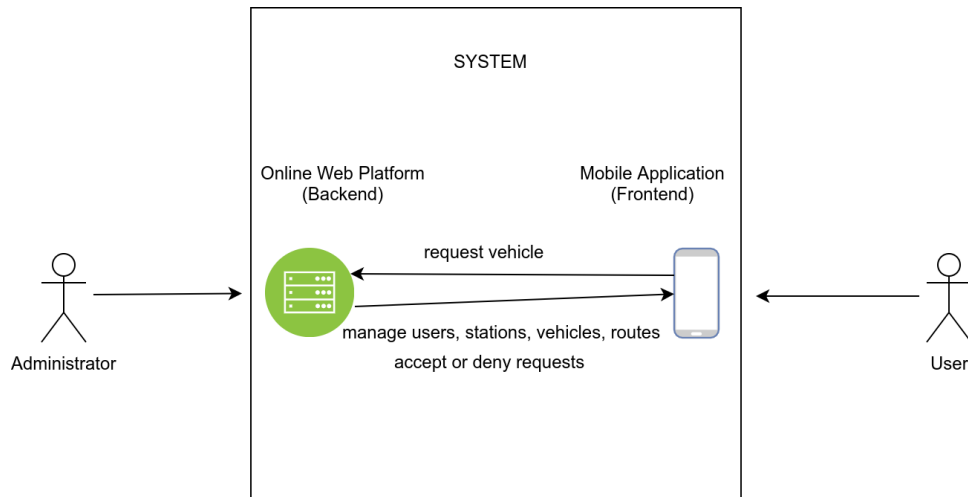


Figure 3.1.1: System overview of the web platform and the mobile application.

access or change data related to those objects. The mobile application is the frontend of the system and is intended for mobile use. Users can perform actions like register, login, see the available stations in their area and send vehicle requests for trips to the web platform. All requests and actions are handled by the platform, which processes the data and sends the appropriate responses, like the user or station data and the status of the user request (accepted or denied). Users can see the status of the vehicle requests in their history page on the mobile application. A system overview of the web platform and the mobile application is depicted in Fig. 3.1.1.

A car sharing company with EVs in their fleet can benefit from our software solution in multiple ways. The web platform can be easily installed on a server. Managing and configuring various components is straightforward, like uploading a map or vehicle characteristics with just two steps: creating and uploading a simple XML file. Users can easily install the mobile application on their Android mobile devices and the system is universal, maintainable and extensible.

We designed, implemented and tested an online web platform for managing various aspects of a car sharing service, and a mobile application for Android devices.

The deliverables of the project are the following:

- source code in zip format of the web application project file
- source code in zip format of the Android application project file
- a video demonstration showing the functionalities of the system

3.2 USE CASES

Use cases are accepted in the industry, as they are a useful method of describing how a business operates, uncovering many scenarios and their pre-conditions and post-conditions. Capturing requirements will be easier after exploring use cases from unified modeling language (UML) diagrams. These diagrams are easy for others to understand and very useful to software engineers, as they help them decide on how to create the classes and objects in object oriented programming languages.

In order to extract all the functional and non-functional requirements we created use cases and treated the proposed system as a black box. We focused on how the system should behave, without making decisions about its internal structure. Our aim and objectives are then translated into a list of *what the system can do* and *what an actor can do*. In our case the actors are the administrators and the users.

The target audience of the online web platform is the administrators. Administrators are responsible for adding, editing and deleting data. They monitor real time information about the system and act on it if needed. In case of an error or a vehicle malfunction, they can mark the vehicle as unavailable or even stop the service of the system. Administrators have the profile of tech savvy people, they have used backend systems before and they are familiar with client-server communication technology.

From the use case UML diagram (Fig. 3.2.1) we see that administrators can: (1) upload a map that contains stations in XML format, (2) upload a list of vehicles in XML format, (3) add, list, update and delete other administrators (name, role),

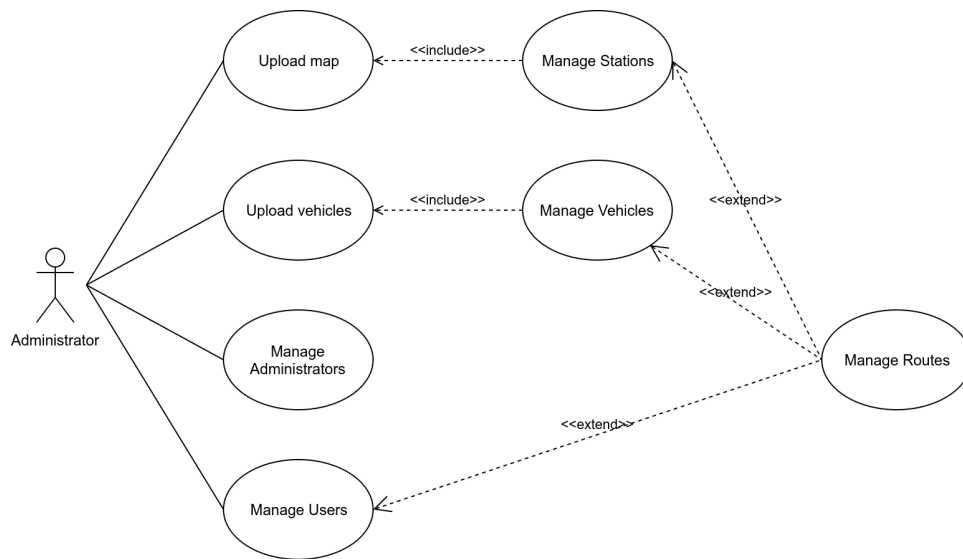


Figure 3.2.1: Administrators use case UML diagram, showing what administrators can do.

(4) add, list, update and delete users (name, gender, date of birth), (5) add, list, update and delete stations (name, latitude, longitude, traffic level), (6) add, list, update and delete vehicles (model, charge level, availability), (7) add, list, update and delete routes (user, start station, finish station, start time, end time, vehicle).

The target audience of the mobile application is the users. Users are the customers of the car sharing company and they rent vehicles for their every day trips in the city. They use the services of the the company, like creating an account, accessing and editing their account data, and making vehicle requests. Users are the everyday people who have an Android mobile device with access to Google Play Store. They know how to install an application and the registration and login processes are very familiar to them.

From the use case UML diagram (Fig. 3.2.2) we see that users can: (1) register with their account details (name, gender, date of birth, username, password), (2) login with a username and password, (3) access their profile details and edit them (name, gender, date of birth), (4) access the history of their vehicle requests (date, time, status), (5) delete the history of their vehicle requests, (6) change the appli-

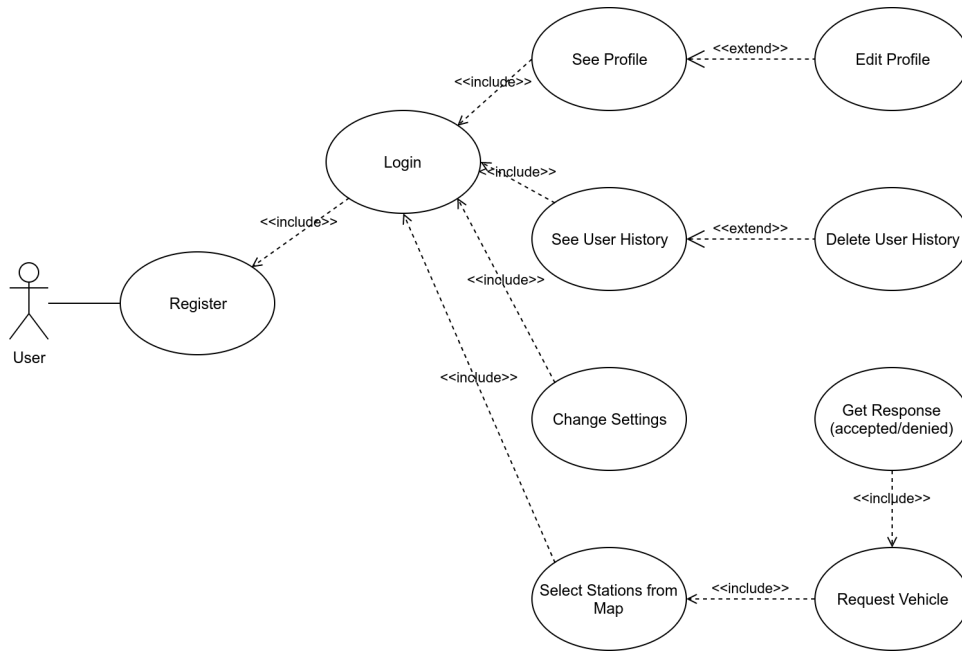


Figure 3.2.2: Users use case UML diagram, showing what users can do.

cation settings (server IP address, port number), (7) select stations from a map (start station, destination station), (8) request a vehicle for a trip (start station, destination station, time), (9) get the response from server regarding whether their request is accepted or denied.

3.3 DATA COLLECTION

To show the web platform and mobile application functionalities in action, we needed to select a city as an example and collect some data, like the number and the locations of the charging stations, and the number of vehicles and their characteristics for a typical car sharing company with EVs. These data are used to create an XML file for loading a map with stations and an XML file for loading the vehicle characteristics in the web platform.

Our research showed that Bristol is an excellent and realistic candidate as a city

to host a car sharing scheme. The City Council in Bristol has already recognized the benefits of car clubs in urban mobility and provided technical and financial support to the Bristol Environmentally Sustainable Transport organization (Cox, 2001). Bristol is also the city that recently shared £40 million with Nottingham, Milton Keynes and London as a part of the "Go Ultra Low City Scheme", so significant investing will be available for EV car sharing support and charging infrastructure (Tietge et al, 2016).

The charging stations and their locations were drawn from *opendata.bristol.gov.uk*. We downloaded the CSV file with the longitude and latitude details and compared them with the most up to date list of Bristol charging points in *www.zap-map.com*. In the final list we kept only the stations with public access and used them in our example to showcase the functionality of the system. We ended up with 27 stations as the potential stations of a new car club with EVs in its fleet and we chose the number of EVs to be 18, a 60% of the total number of stations, as this is a typical ratio in car clubs in the UK.

3.4 REQUIREMENTS ANALYSIS

The requirements for this project are the functional, that define the function of a system or its component and the non-functional, that specify criteria like security and performance. We will first focus on the requirements for the web platform and the functionalities for administrators and then on the requirements for the mobile application and the functionalities for users.

Functional requirements for administrators include all actions described in the administrators use case UML diagram. Additionally, the web platform should also accept requests from the mobile application, compute all the factors that affect the number of people that will be served in a given time period, handle user requests from the mobile application and decide if it will accept or deny them. Other data that will be sent to the mobile app include user account and station details. Non-functional requirements focus on security. This is a major concern, as many users will interact with the system. For this reason we had to make sure that all pages are

protected from malicious actions like SQL Injection attacks and that passwords must be hidden (hashed and "salted") to protect the administrator and user accounts. Moreover, the performance should be fast and reliable, allowing for an uninterrupted service to administrators and users (Table 3.4.1).

TYPE	ACTIVITY	PRIORITY
Functional	Login with a username and password	1
Functional	Add/list/update/delete administrators	1
Functional	Upload a map with stations in XML	1
Functional	Upload a list of vehicles in XML	1
Functional	Add/list/update/delete users	1
Functional	Add/list/update/delete stations	1
Functional	Add/list/update/delete vehicles	1
Functional	Add/list/update/delete routes	1
Functional	Handle requests from users	1
Functional	Send request status to mobile app	1
Functional	Send user account details to mobile app	1
Functional	Send station details to mobile app	1
Security	Protect against SQL Injection	1
Security	Store only hashed values of passwords	1
Performance	Fast load and response rate	1
Performance	Offer uninterrupted service to users	1

Table 3.4.1: Functional and non-functional requirements for administrators and the web platform.

Functional requirements for users include all actions described in the users use case UML diagram. Additionally, the mobile application should send requests to the web platform and accept the status of the request. It should also access user account and station details from the database. Non-functional requirements focus on security. User authentication must be safe by transmitting only encrypted messages to the platform. The performance and response of the mobile application should be fast, offering a pleasant user experience to its users (Table 3.4.2).

TYPE	ACTIVITY	PRIORITY
Functional	Register with account details	1
Functional	Login with a username and password	1
Functional	Access/edit profile details	1
Functional	Access/delete history of requests	1
Functional	Change the application settings	1
Functional	Get station details from platform	1
Functional	Select start/finish stations from a map	1
Functional	Select specific time for a request	1
Functional	Request a vehicle for a trip	1
Functional	Get request status from the platform	1
Functional	Save request status to history	1
Security	Send encrypted messages to the platform	1
Security	Authenticate users when requesting data	1
Performance	Fast load and response rate	1
Performance	Offer uninterrupted service to users	1

Table 3.4.2: Functional and non-functional requirements for users and the mobile application.

3.5 TECHNOLOGY RESOURCES

In order to achieve the goals of an easy maintainable and extensible software solution, we examined various web application frameworks. Laravel, with modular packaging and relational database mapping, CakePHP with great plugins and community support, and CodeIgniter, powerful with a very small footprint, are very popular PHP frameworks. Java based frameworks used for web development are Spring MVC, for modern enterprise applications, JavaServer Faces, for building great user interfaces, and Struts, extensible with plugins. For the implementation of the web platform we chose the Java Spring MVC framework, as it comes with important technologies that helped us accomplish our objectives.

Java Spring is a very popular open source development framework. It allows developers to create easily testable and reusable enterprise Java applications with excellent performance. It simplifies development with bundled modularity and the

use of Plain Old Java Objects (POJOs). Simpler code becomes easier to test and maintain. Some of the most important characteristics of the Java Spring framework are Inversion of Control (IoC) and Dependency Injection (DI). IoC enables Spring to control the flow of the program and make calls to our custom code. This feature makes it easier to switch between different implementations and offers greater modularity and easier testing by isolating dependencies. IoC can be achieved by various mechanisms, like Dependency Injection. With DI, objects are injected into other objects by the framework itself.

The Spring framework has great community support and comes ready with very important technologies that help developers create better applications. Some of those technologies are: Spring MVC, Spring Security, Spring Batch, Spring JDBC and DAO, Spring ORM, Spring AOP. Spring MVC is a web framework built on the Servlet API that follows the Model View Controller design pattern. Spring Security is used for protecting applications with extensible authentication and authorization support. Spring Batch simplifies and optimizes the processing of high volume batch operations, while Spring JDBC and Data Access Object (DAO) modules provide translations from specific API exceptions to the DAO hierarchy. The Spring Object Relational Mapping (ORM) module provides an abstraction layer for other APIs that map objects to tables in a relational database, such as Hibernate. Lastly, Spring Aspect Oriented Programming (AOP) offers another mode of modularity, the "aspect", allowing developers to decouple code that implements functionality that should be separated.

For the Android application development we had to ensure proper data exchange between the application and the web platform on the server. We found two options that can serve this purpose: implementing a RESTful web service or using WebSockets technology. A RESTful web service is based on the REpresentational State Transfer (REST) technology, an abstraction for creating APIs for applications in a standardized way. With REST endpoints, an application can receive requests and respond to them. Requests come from a client that uses common HTTP verbs like GET, POST, PUT, DELETE for retrieving, submitting, updating and deleting data. Responding to requests is usually realized by sending data

back to the client in XML or JSON format.

WebSockets is a protocol between a client and a server that runs over a persistent TCP connection. A TCP socket connection can be opened, and bidirectional and full-duplex messages can be sent between the client and server and then the connection can be closed. The opening and closing creates overhead, but the connection offers reliability in the exchange of messages. The major advantage of creating a connection is that developers do not need to handle a server side service that is always listening and ready to perform specific tasks based on extra criteria from a client request. Instead, the TCP connection handles the delivery of the data and then it is up to the application layer to apply certain security criteria before sending the appropriate response message to the client (Table 3.5.1).

CHARACTERISTICS	REST SERVICE	WEB SOCKETS
What is sent and received	HTTP messages	TCP messages
Creates connection (state)	No	Yes
Full-duplex communication	No	Yes
Managing resources is defined	Yes	No
Scaling capability	Horizontally	Vertically
Performance	Good	Slightly better

Table 3.5.1: Characteristics of different data exchange methods between the platform and the mobile application.

Mainly because of the different scalability and the stateful/stateless capabilities of these exchange methods, we incorporated both of them, but in different situations. Communication via RESTful web services is used for registering, logging in, and sending station information to the mobile application. When the user sends the request for a vehicle to the server through the app, we use WebSockets technology in order to have a stable connection until the platform sends its response back to the mobile app. Sending messages to an IP address and port number with the TCP protocol is slightly faster and more efficient than web services, so this method

fits better to the vehicle requests functionality.

The software resources for the design and implementation of the project are: (1) Eclipse Java EE IDE for Web Developers Oxygen Release for developing with Java, (2) Tomcat Server 9.0 to run the Spring MVC project on the localhost, (3) MySQL Workbench 6.2 to create and manage the platform database, (4) Android Studio 2.3.3 for developing the Android application, (5) Atom 1.15 for fast search in directories and refactoring code, (6) GitHub as a version control software, (7) Photoshop for creating and editing graphics and images, (8) Google Drive for manual backup (e.g. after every milestone).

The hardware resources for this project are: a laptop with latest I7 7700HQ CPU 2.80Ghz and 12GB of DDR4 RAM for developing the system in our local machine, an online server for hosting the web platform, and an Android device for testing purposes (we used a Samsung Note 3 mobile phone). The commercial resources needed for bringing the project online and making it available to users are: domain name from GoDaddy (cost of 12 EUR), hosting service in DigitalOcean (cost of 100 EUR for one year of hosting), and Android developer account (cost of 20 EUR) to publish the application to Google Play marketplace. Commercial resources are optional and should be acquired by the car sharing company.

4

System Design

4.1 ARCHITECTURE

In the previous chapter we presented the functionalities of administrators and users in the web platform and mobile application respectively. We will now examine the architecture of the system and how the parts communicate and exchange data with each other. Those functionalities fulfill the basic operations of a car sharing company that offers EVs to their clients.

The web platform is installed on the server and it consists of the web pages presented to the administrators and a MySQL database to manage all the data. Administrators interact with the platform using web pages designed for specific functionalities. So, the interface design of the pages allow them to login, upload a map with stations, upload vehicles details and manage administrator and user accounts,

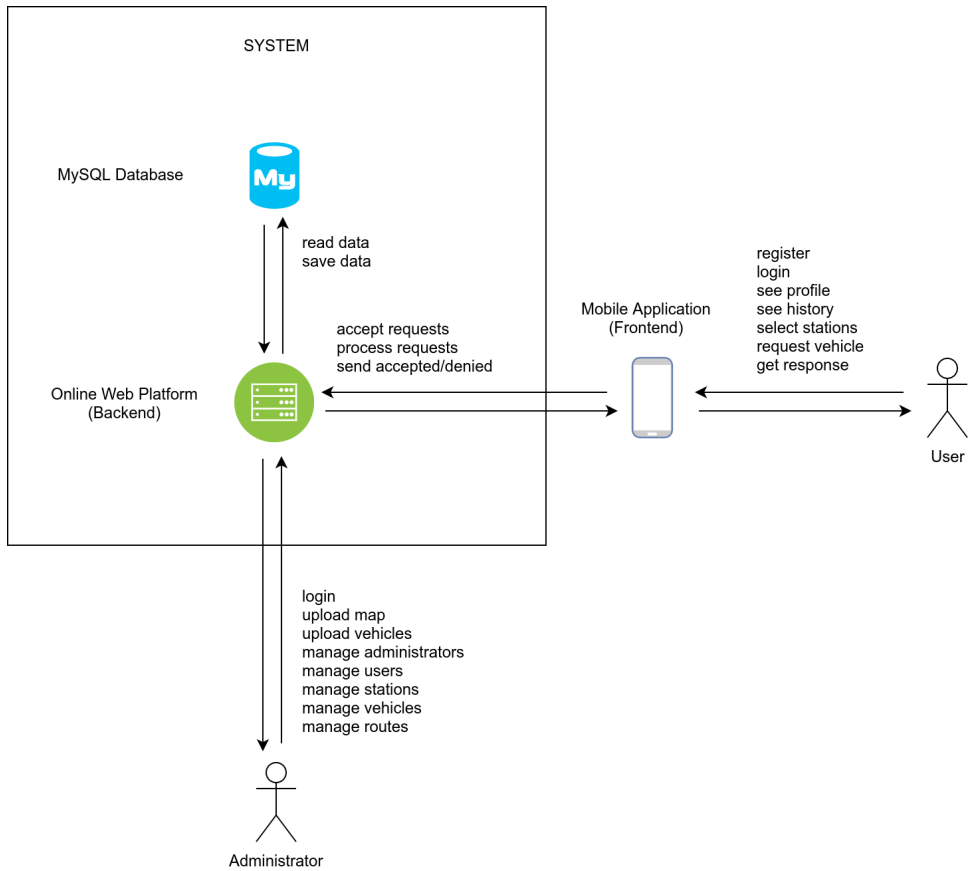


Figure 4.1.1: System architecture with web platform, database and mobile application.

stations, vehicles and routes.

All operations submitted through the interface of the pages are implemented as transactions in the MySQL database. MySQL database is made up of tables with attributes and values. Rows of data are saved within these tables, so administrators can manage the saved objects. The mobile application lets users register, login, access their profile, their request history, select stations and request a vehicle for a trip. Creating an account, and accessing account and station details require access to the MySQL database on the server (Fig. 4.1.1).

Information exchange with the database is done via a RESTful web service that authenticates users and allows them to access, create and edit data on the database. Messages for vehicle requests use WebSockets, so the server listens to a specific port for incoming TCP messages. Those messages will be encrypted and sent from the mobile device to the server through a TCP connection, making the communication safe and reliable. The platform will then apply a decision algorithm that depending on the data in the database at the current time, it will analyze all future routes, station and vehicle information and accept or deny the user request. The status of the request (accepted or denied) will be sent back to the mobile application as an encrypted message through the TCP connection.

4.2 WEB PLATFORM DESIGN

The web pages of the platform let the administrators perform the functionalities discussed earlier. The login page allows administrators to login with their username and password. On successful login, they are redirected to the Dashboard Panel. The Dashboard Panel shows route data and contains buttons to update, delete or add a new route. The stations are shown as pins on a Google map and buttons allow administrators to start or stop the service.

The Manage Admins and Manage Users pages show administrator and user data respectively, and contain buttons to update their existing details, delete them or add new administrators and users. The Manage Stations and Manage Vehicles pages offer the same functionalities, but they also allow administrators to upload

stations and vehicle details from files in XML format. For each object there are two secondary pages, the Add New page and the Update page to provide secondary functionalities like adding and updating objects. All pages of the platform, their activities and their parent page are presented in Table 4.2.1.

PAGE	ACTIVITY/ACTION	PARENT PAGE
1. Login Page	Administrator login	-
2. Dashboard Panel	Access routes data/delete routes	1
3. Add New Route	Add a new route	2
4. Update Route	Update existing route data	2
5. Manage Admins	Access admins data/delete admins	1
6. Add New Admin	Add a new admin	5
7. Update Admin	Update existing admin data	5
8. Manage Users	Access users data/delete users	1
9. Add New User	Add a new user	8
10. Update User	Update existing user data	8
11. Manage Stations	Access stations data/upload XML	1
12. Add New Station	Add a new station	11
13. Update Station	Update existing station data	11
14. Manage Vehicles	Access vehicles data/upload XML	1
15. Add New Vehicle	Add a new vehicle	14
16. Update Vehicle	Update existing vehicle data	14
17. Simulation Panel	Access simulation data/upload XML	1
18. Live Charts	Access live charts and statistics	1

Table 4.2.1: The pages of the web platform and the functionalities performed by administrators.

The Graphical User Interface (GUI) of the web platform consists of elements that are similar and shared between the pages. The Login page contains the logo and a small login form. All other pages share the same design. At the top there is a horizontal bar that on the left has the title and on the right a personalized welcome message to the administrator. In the left vertical area we have the logo in

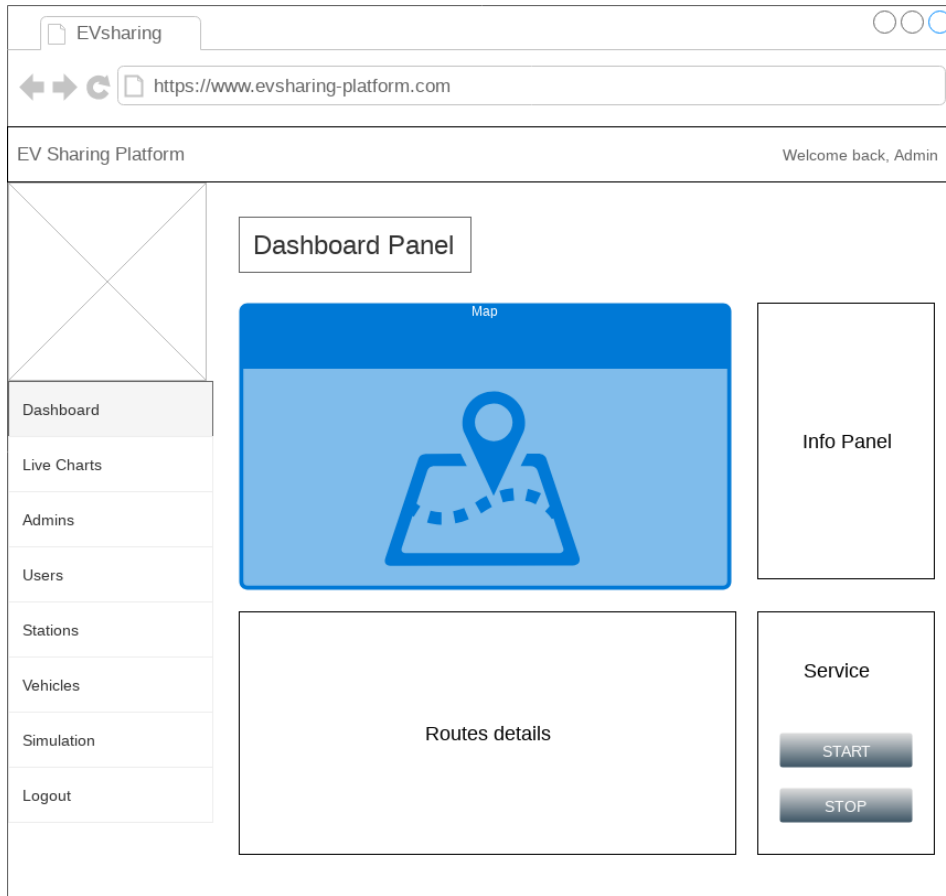


Figure 4.2.1: Mockup design for the Dashboard page. Simulation page shares a similar design.

rectangular form and just below it, the navigation options link to the most important pages: Dashboard, Live Charts, Admins, Users, Stations, Vehicles, Simulation and Logout.

The center area is specific to each page and contains the title (the Map area for Dashboard and Simulation pages only, as seen in Fig. 4.2.1) and the Details area. The Details area is bigger in Admins, Users, Stations and Vehicles (as seen in Fig. 4.2.2), with rows that correspond to each object in the database. The right area on all pages contains an Info Panel, where some useful information are provided for

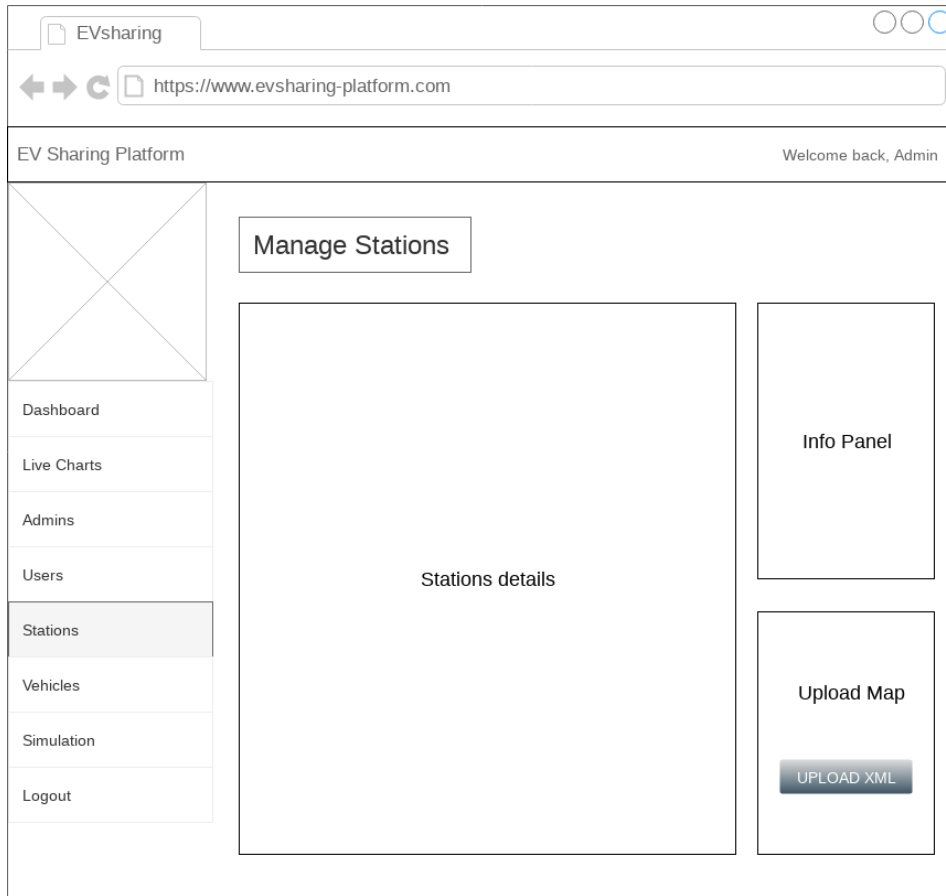


Figure 4.2.2: Mockup design for the Stations page. Admins, Users and Vehicles pages share a similar design.

various actions within the page. In the Dashboard and Simulation pages there are buttons for starting and stopping the service and the simulation, and in Stations and Vehicles pages an Upload button allows administrators to upload an XML file with station or vehicle details respectively.

4.3 MOBILE APPLICATION DESIGN

The Android mobile application consists of a set of screens. The Main Login screen allows users to login with their username and password. If they do not have an account, they can click the Register button to create a new account with their details. On successful login, they are redirected to My Profile screen. The Profile screen shows their details and contains buttons to update them and to move on to the next screens, like Open Map and User History.

The Open Map screen initiates the first step for requesting a vehicle and the map is shown with the stations as pins. Users can select the start and finish stations and on the second step, on the Request screen they can select a specific time and send the request to the platform. The platform will process all current data and decide whether to accept or deny the request. The mobile application gets the status of the request from the server and saves it to the device. The User History screen contains the history of requests and the Settings screen allows users to change the application settings. All screens and activities are presented in Table 4.3.1.

SCREEN	ACTIVITY/ACTION	PARENT SCREEN
1. Main Login	User login	-
2. User Registration	Add a new account	1
3. My Profile Screen	Access/edit profile details	1
4. User History	Access/delete history of requests	3
5. Open Map Screen	Get and select start/finish stations	3
6. Request Screen	Send request, get and save response	5
7. Settings Screen	Change the application settings	1

Table 4.3.1: The screens of the mobile application and the functionalities performed by users.

The GUI of the mobile application consists of elements that are similar and shared between the screens. The Main Login screen contains a simple login form

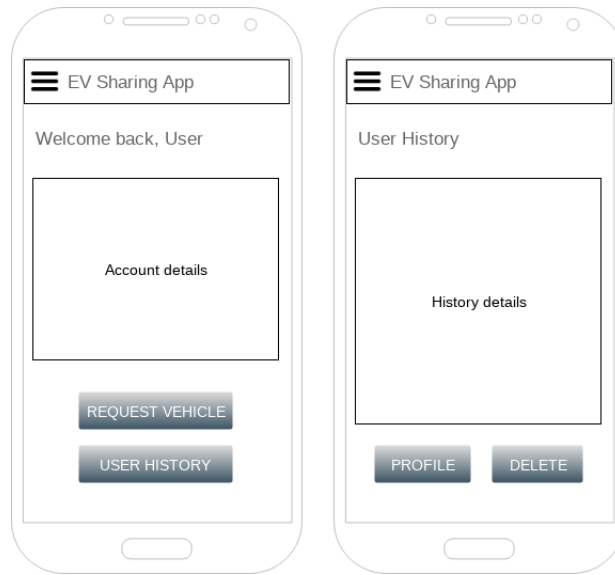


Figure 4.3.1: Mockup designs for My Profile and User History screens.

and a button for creating a new account. At the top of all screens there is a horizontal bar with a title. On the left there is an icon that opens up the mobile menu. The navigation drawer is the panel that shows the logo in rectangular form and just below it, the navigation options link to the most important screens: My Profile, Open Map, Request Vehicle, User History, Settings and Logout. The navigation menu is hidden when the user touches the rest area of the application (Fig. 4.3.2).

The center area is specific to each screen and contains the title, the Details area and one or more buttons. The Profile screen has buttons that let users open the map, request a vehicle and access their request history. User History has a button to allow navigation back to My Profile and a button to delete the request history that is saved on the device (Fig. 4.3.1). The Open Map screen contains a Map area with the stations (Fig. 4.3.2), and on the Request screen there are buttons for sending the vehicle request and for navigating back to My Profile.

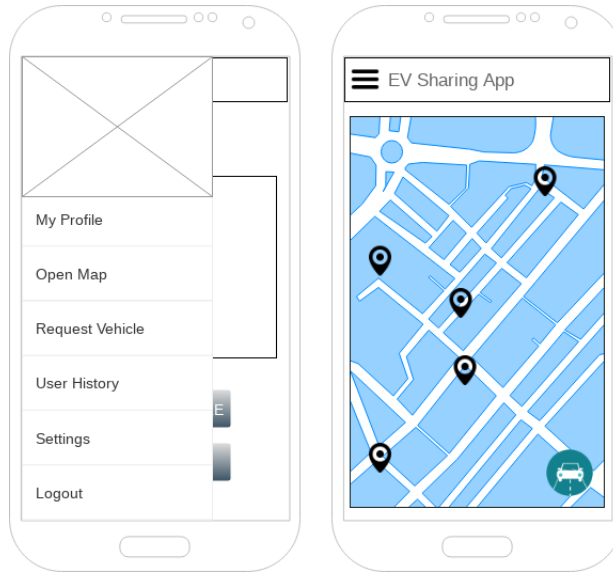


Figure 4.3.2: Mockup designs showing the navigation drawer menu and the Open Map screen.

4.4 DATABASE DESIGN

The web platform uses a database to manage objects or entities and their characteristics. Those entities are: administrators, users, stations, vehicles, routes and simulations. The database consists of tables that represent each entity and holds a number of attributes. Every object is saved as a row of those attributes and their values. Primary keys are used to uniquely identify a saved object and foreign keys are used to refer to a set of attributes from another table. Primary and foreign keys form the relationships between the tables.

The attributes of the user table are the username, password, name, gender and date of birth. The administrator table holds the username, password, name and role. There are two roles for administrators, the admins that can create new admins and the moderators that can not add or remove admins from the database. The attributes of the station table are the name, longitude, latitude and traffic level. We have defined the traffic level as a factor that influences how frequent the routes are at specific stations, and how much time and energy a vehicle needs to fulfill

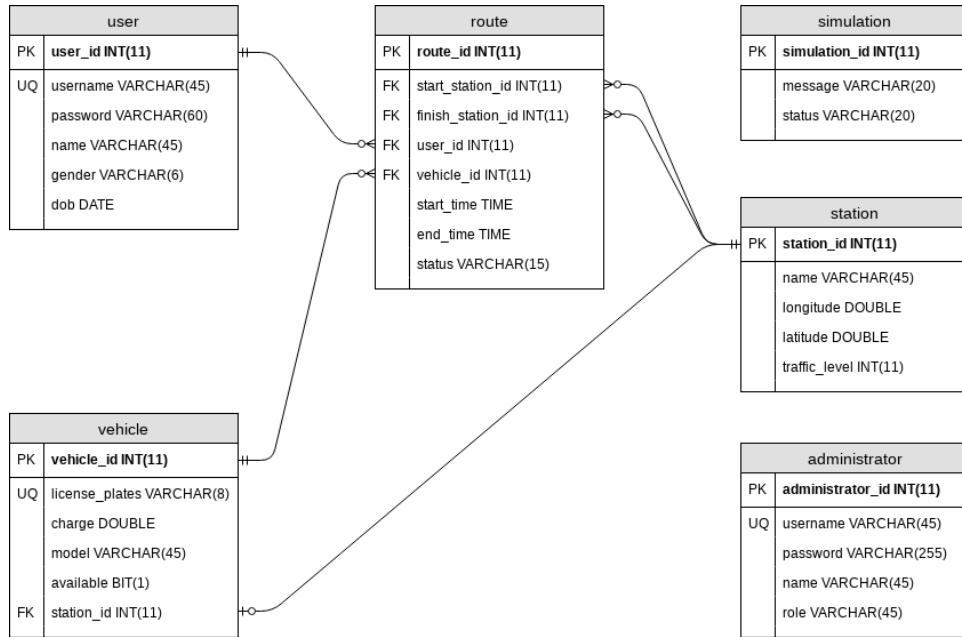


Figure 4.4.1: Entity relationship diagram of the database.

them. There are 5 traffic levels, where level 5 stations are located very close to the center of the city. As we move further, we encounter stations with traffic levels 4 to 1. The vehicle table holds the license plates, charge level, model, availability and station id. Charge level is a percentage of the energy of a vehicle in a given time, the availability is *true* when the vehicle is available and is set to *false* in case of a malfunction. The station id has the id of the station that the vehicle is currently located.

The attributes of the route table are the start station id, finish station id, user id, vehicle id, start time, end time and status. These attributes are used to describe the data of a route (for example user₁ requests a route from station A to station B at 16:00). If the status of the request becomes accepted by the system, it will calculate the end time and assign a specific vehicle to this route. The status attribute gets the value of "Accepted" or "Denied", so the system saves the requests that have been denied as denied routes and the valid routes as accepted routes.

The simulation table is used when we want to load a batch of requests from an XML file, thus simulating a specific amount of user requests for vehicles. By loading a simulation file, the system processes all data and decides on the status of each request. The simulation table holds the message of each simulation in the same form as if it had been submitted by the mobile application, along with the status. The status of each simulation is first set to "Ready" and when the system decides about the status, it is set to "Processed".

There are five relationships between the entities: (1) one and only one user can be part of zero or more routes, (2) one and only one vehicle can be part of zero or more routes, (3) one and only one station can be part of zero or more routes as a start station, (4) one and only one station can be part of zero or more routes as a finish station, (5) one and only one station can be assigned to zero or one vehicle (Fig. 4.4.1).

5

Implementation

5.1 DEVELOPMENT PHASES

We have grouped the required tasks for the development of the project into the following five phases of development: Phase I (web platform), Phase II (mobile application), Phase III (platform scheduler algorithm), Phase IV (platform decision algorithms) and Phase V (Testing and evaluation). Each phase is an iteration cycle that includes analysis, design, implementation and testing procedures (Fig. 5.1.1).

In Phase I we developed the web platform. Specific requirements were analyzed in more details, the design was revised and we implemented the platform starting from small components. Before moving to bigger components, we performed manual testing and when adding components, we always tested the communication between them. In Phase II we followed the same steps for developing the

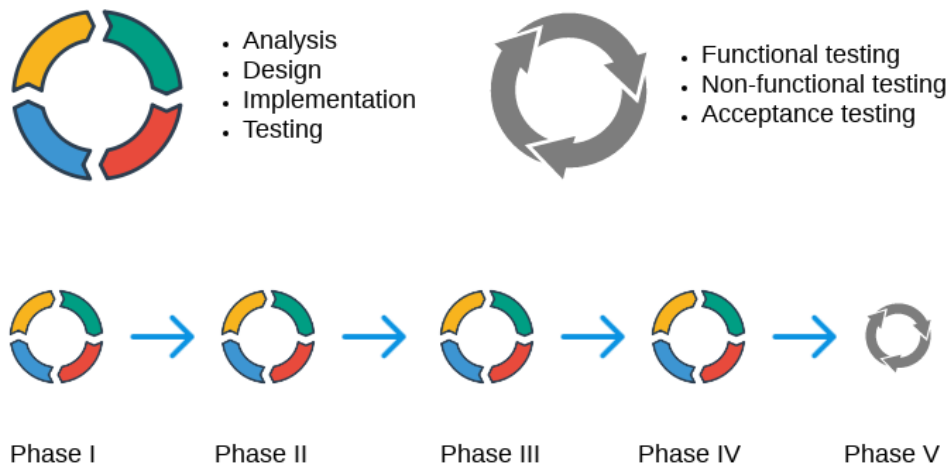


Figure 5.1.1: Phases of development of our software solution.

mobile application.

Phase III was about research and development of the platform scheduler. The scheduler is responsible for making the appropriate calculations of the vehicles directions, charging levels and their distances to nearby stations in order to predict future routes and the efficiency of the system. In Phase IV we created two algorithms for deciding whether to accept or deny a vehicle request. The "short mode" algorithm takes into account the future routes of the vehicles in the start station and decides if these vehicles are free or not, and the "long mode" algorithm considers the future routes of other vehicles and if they can substitute a current "locked/assigned" vehicle to make it free for use.

In Phase V functional and non-functional testing was performed using black box test design techniques to detect possible defects. White box testing was used to check the thoroughness of testing. User acceptance testing was also important, in order to gather user feedback and conclude if the system fulfills its needs. More details about the project Gantt chart and the project plan can be found in Appendix sections 9.1 and 9.2.

5.2 WEB PLATFORM IMPLEMENTATION

Spring MVC is a request driven framework. All incoming requests pass through a central servlet, the front controller, that delegates the requests to a specific controller. This controller handles the request and builds the appropriate model, which encapsulates the application data. Then, it sends the model to the view for rendering the data and generating the HTML output for the client's browser. The process of building the model includes the use of a service class that delegates calls to a suitable DAO class. DAO classes initiate the actual database transactions with the help of entity classes and object relational mapping modules (Fig. 5.2.1).

In our application we use a package named Controllers with all controller classes. Each controller accepts as input user data and with the help of services classes, it updates the model and passes it to the appropriate view. Then, a JSP page generates HTML output with the updated model data. We have the following controller classes that handle the user request when accessing a specific web page: HomeController, AdministratorController, UserController, StationController, VehicleController, SimulationController, LiveChartsController. The controllers that handle the RESTful web service for allowing access to user accounts and stations details from the mobile device are: UserRestController and StationRestController.

The Repositories package contains interface DAO classes and their implementations. Those classes read and write data to the database by accessing the POJOs in the Entities package that map objects to specific tables in the database. We have the entities classes: Administrator, User, Station, Vehicle, Route, Simulation and each one has its corresponding class in the Repositories package that calls its methods. We used Hibernate as an object-relational mapping framework.

The services implementations inside the Services package delegate requests to the suitable DAO classes and offer a specific functionality that controllers classes need. The classes in this package are the following: AdministratorService, UserService, StationService, VehicleService, RouteService, SimulationService and they are used to list, save, update and delete administrators, users, stations, vehicles, routes

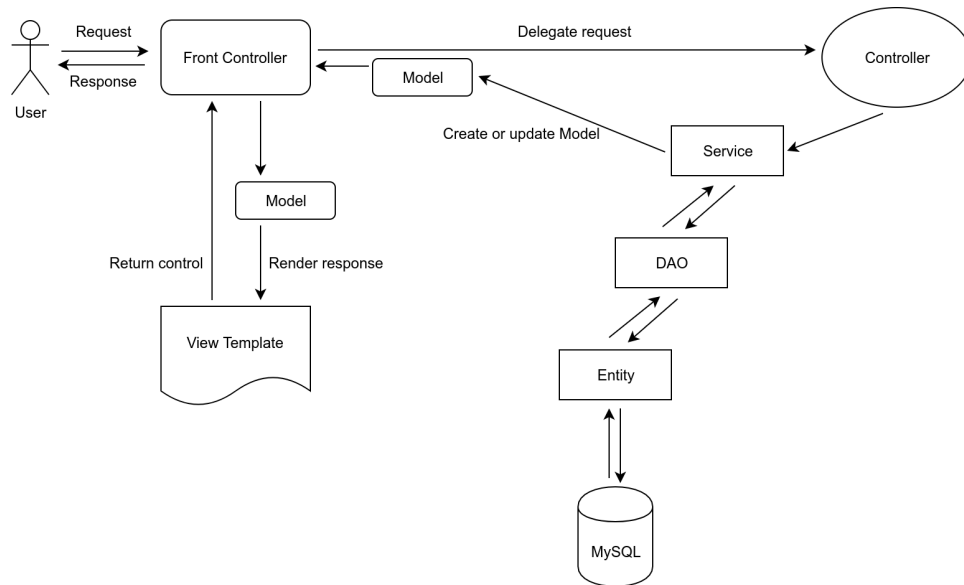


Figure 5.2.1: Overview of Spring MVC architecture for the web platform.

and simulations respectively. LoadMapService, LoadVehiclesService, LoadSimulationsService are used for parsing the uploaded XML files and saving stations, vehicles and simulations into the database. The PopulateDropdownsService creates the data for users, vehicles and stations dropdown lists. ProcessRequestServiceShort and ProcessRequestServiceLong classes implement the "short mode" and "long mode" algorithms for processing the requests and RemoteConnectionService is used to create a TCP socket for listening to incoming TCP messages.

The RESTful web service allows access to user accounts and stations details from the mobile device, so we need an authentication and authorization framework to restrict this access to registered users that use our mobile application. In order to secure the web service we used OAuth2, where third-party applications can obtain limited access to an HTTP service on behalf of a resource owner. The resource server hosts the protected resources and responds to client requests for accessing them. The authorization server issues access tokens to the client after authenticating the resource owner and obtaining authorization.

The Security package contains the classes for configuring the authentication and

authorization implementation for the RESTful web service. Spring Security helps us integrate OAuth2 and it also filters access to specific web pages in order to protect the administration panel. So, we do not only use session variables to restrict access from not logged in users that try to access the web platform administration pages, but we also check their role when they login and filter their page requests according to custom rules specified in a Spring Security class. The classes in this package are: `AuthorizationServerConfiguration`, for configuring the authorization server, `ResourceServerConfiguration`, for configuring the resource server, and `SecurityConfiguration`, for protecting the administration pages and allowing only admins and moderators to access them.

The `Utils` package contains some helper classes used by services classes. `MyTasklet`, configured as a Spring Batch job, is used to predict future routes and the efficiency of the system. This class is essential for the live charts functionality of the web platform. Future movement of vehicles, charge levels and routes need to be predicted in small intervals, so if a user requests a vehicle in a later time, all data will be recalculated and the live charts will be updated accordingly. We chose to execute `MyTasklet` as a Batch job every 10 seconds. `QuartzTaskScheduler` is used for executing the Spring Batch job. `ProcessModeHelper` and `SocketConnectionHelper` are just helper classes for accessing the process mode and client socket variables. `RemoteConnectionHandler` class handles TCP connections when the system processes the user request. `StrongTextEncryptorHelper` class is used to encrypt and decrypt text, in our case the TCP messages between the platform and the mobile application, with a master password. The class diagrams for the web platform can be found in Appendix section 9.3.

Administrators use the web platform to manage users, stations, vehicles, routes and requests from the users. In order for the system to be able to accept and respond to requests, station and vehicle data should be uploaded, a process mode should be selected and the service should be started. Then a TCP socket is created, listening for incoming requests as TCP messages. For each mobile client that communicates with the server, a new TCP connection is created. When messages are received, they are processed with the selected mode and the responses are sent

back to the clients and saved in their User History.

Web server setup is essential in order to create a live environment and test the web platform. We purchased a Virtual Private Server (VPS) from DigitalOcean. We installed Debian 8.9 as the server OS, as Debian releases are very stable and they are widely used in production environments. We upgraded to the latest packages and then installed and secured MySQL Server. We created the same user account that we had when developing the database in MySQL Workbench in our local machine, exported an .sql dump of our locally developed database and imported it on the sever. The next step was to install Java 8 Development Kit (JDK) and Java Cryptography Extension (JCE) Unlimited Strength, which is needed for TextEncryptor to work. Finally, we installed and configured Tomcat Server 9.0 on the server and uploaded the Web Application Archive (WAR) file of the project that we developed with Eclipse Java EE IDE in our local machine.

5.3 MOBILE APPLICATION IMPLEMENTATION

Implementing the Android application includes creating the layout XML files with the appropriate graphic elements and controls for each Activity. Activity classes use their corresponding views from the layout files and handle the data input and processing. The communication with the web platform is realized via a REST client for accessing user accounts and station details, and a TCP connection for sending requests and receiving responses.

We organized the classes of the mobile application into four packages: Base, Evsharingapp, Entities and Utils. The Base package contains the BaseActivity class, which is the base for all Activity classes since they inherit its attributes and methods. BaseActivity provides the functionalities for the navigation drawer and toolbar. It handles clicks on navigation items and starts the corresponding Activity.

The Evsharingapp package contains the Activity classes. MainActivity creates the first screen shown to the user with the login form and the Register button. AsyncTask is used to perform background operations and show the results on the UI thread, without affecting the main thread. Such background operations include

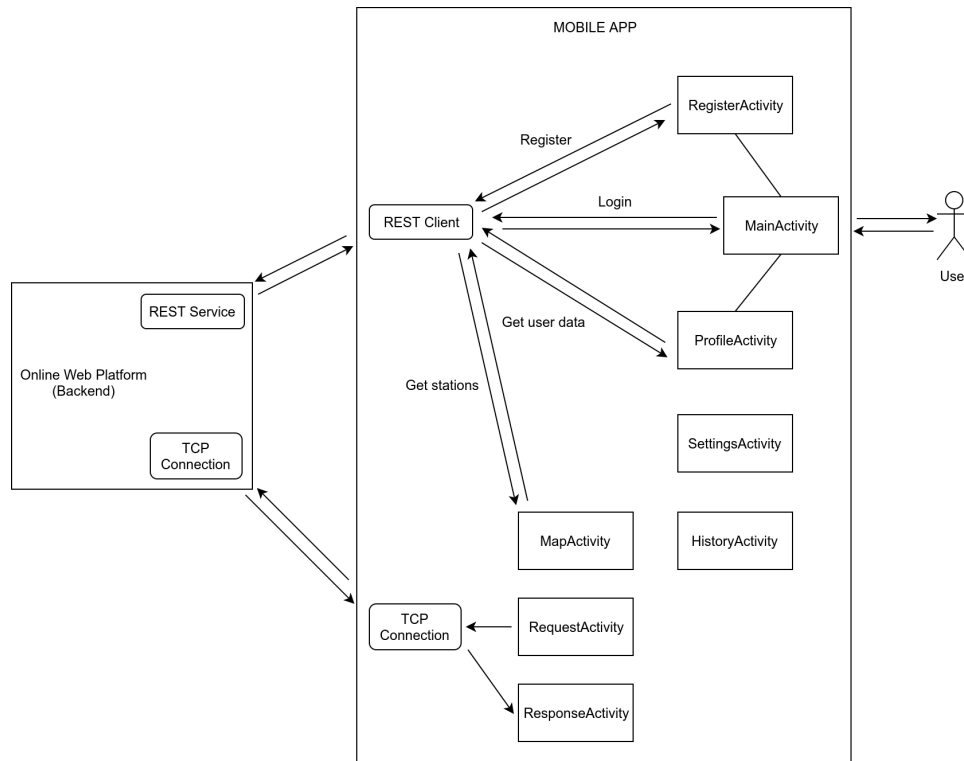


Figure 5.3.1: Interactions between Activity classes, and the communication channels between components of the system.

all communication functionalities with the server. RegisterActivity allows the user to register a new account and the ProfileActivity gets the account details from the server and presents them on the screen using the REST client (Fig. 5.3.1).

The MapActivity gets the station details and shows them as pins on the map. When the user selects the start and finish stations, the RequestActivity loads a new screen with the station names in dropdown elements and a control to select the time. When the user clicks on the Send Request button, a TCP message with the encrypted request details is sent to the server. The encrypted text contains the user id, the start station id, the finish station id and the time. When the response is received, the ResponseActivity shows the request status (accepted or denied) to the user, which is then saved in a local database on the Android device. HistoryActiv-

ity loads the saved requests from the database into a properly formatted webview component. The SettingsActivity allows the user to configure the IP address and port number of the server.

The Utils package contains the classes responsible for the communication via the RESTful service. AuthTokenInfo represents the token object required for the OAuth2 authorization procedure. The SpringRestClient class implements the authentication and authorization by requesting a token from the resource owner. The authorization server checks the mobile application credentials, issues a token and then the application can access the resources on the server, in our case the user accounts and station details. The data retrieved from the server are saved as *user* and *station* objects with the POJOs in the Entities package.

The DateDialog and TimeDialog classes in the Utils package are used to create the date and time dialogue boxes. MyTextEncryptor encrypts and decrypts the TCP messages between the platform and the mobile application, using the same master password used on the corresponding class of the platform. The Helpers class includes functionalities related to the management of the database on the Android device, like loading, updating and deleting rows (used in request history with status details). The Utils package also includes methods for validating input data in the login, register, profile and request forms and the Global class contains attributes available globally to all classes, such as the IP address and port number of the TCP server, the current user, the SQLite database and the database file objects. Mobile application class diagrams can be found in Appendix section 9.4.

The workflow of a vehicle request by a user consists of the following steps: (1) the user logs in to his account from the mobile application, (2) he selects Open Map from the navigation menu, (3) he selects the start station and the finish station from the pins on the map, (4) he selects a specific time and clicks on the Send Request button, (5) the encrypted message is sent to the server via the TCP connection, (6) the web platform decrypts the message request, (7) the selected process mode determines which decision algorithm will be executed ("short mode" or "long mode"), (8) a route is created and saved into the database with the appropriate status, (9) the request status is sent back to the mobile application.

5.4 THE PLATFORM SCHEDULER

The purpose of the platform scheduler is to provide a way to execute a Batch job in small intervals that predicts the future movement of vehicles, the charge level and the routes of each vehicle within a given time period (e.g. a day). For every incoming vehicle request, the prediction data have to be recalculated, considering the result of the decision algorithm of the system, which is to accept or deny the request. This way, the live charts and statistics are updated every 10 seconds, which is the interval that the scheduler algorithm is executed.

The scheduler algorithm creates and updates three HashMaps: `stationsMap`, `vehiclesMap` and `chargeMap`. The `stationsMap` has the station id as an integer key and an array of integers as the value, that represent the current sum of vehicles in the station. The `vehiclesMap` has the vehicle id as an integer key and an array of integers as the value, that represent the current station of the vehicle with the station id. The `chargeMap` has the vehicle id as an integer key and an array of doubles as the value, that represent the current charge level of the vehicle. We assume that the system will operate for 16 hours within a day in order for low level vehicles to be charged and checked for possible malfunctions. This equals to 192 5-minute segments, so to predict the future values in the HashMaps, we used a size of 192 for all arrays. We can predict, by getting the value of the array at the index that corresponds to a specific 5-minute segment: (1) how many vehicles will be available in a specific station, (2) the current station of a specific vehicle, (3) the current charge level of a specific vehicle.

The platform scheduler algorithm is implemented in `MyTasket` class. First, the vehicles are assigned to stations with the highest traffic level. Then, the `stationsMap`, `vehiclesMap` and `chargeMap` are created for every station and every vehicle. We update the HashMaps according to the starting positions of the vehicles and if the simulation mode is active, we get all simulation requests and process them according to the selected process mode. Next, we update the HashMaps with the appropriate details from the accepted routes in the system. For each route, the end time and charge cost are calculated, along with the number of vehicles on stations,

the current station of the vehicles and their charge level, by updating the values of the arrays. Now, the stationsMap contains key-value pairs for all stations, and vehiclesMap and chargeMap contain key-value pairs for all vehicles.

Additionally, three arrays of integers are used to calculate the total number of requests, the accepted requests and the denied requests in the system. The size of these arrays is also 192, representing the total 5-minute segments within a day. The HashMaps and the three arrays are then sent to the LiveChartsController class, which sends the data to the Live Charts web page. The Live Charts page shows five statistical diagrams: the efficiency rate, which is the number of accepted user requests against the number of denied user requests, a chart showing the accepted and denied requests over time, the vehicles allocations in stations diagram, the vehicles charge levels over time diagram, and a table showing the current positions of the vehicles over time.

5.5 SHORT MODE DECISION ALGORITHM

The most important element in the evaluation of the system is to ensure that the highest number of users will be served within a day. We have developed and implemented two decision algorithms that process the user requests and decide to accept or deny them, in order to maximize the number of accepted requests against the denied requests. We will examine those two algorithms and see what are their key differences.

The Short mode algorithm gets the request message and extracts its attributes, the user id, start station, finish station and start time. The trip duration in minutes, and the charge cost for the request trip are calculated. Next, we get the vehicles that are currently in the start station and add them to the candidateVehiclesList. We also get the vehicles that finish at the start station before the start time, and add them to list. If the candidateVehiclesList equals to zero, the "result" string is set to "Denied", else we check if the candidate vehicles have future routes. We add the vehicles without future routes, to the betterCandidateVehiclesList.

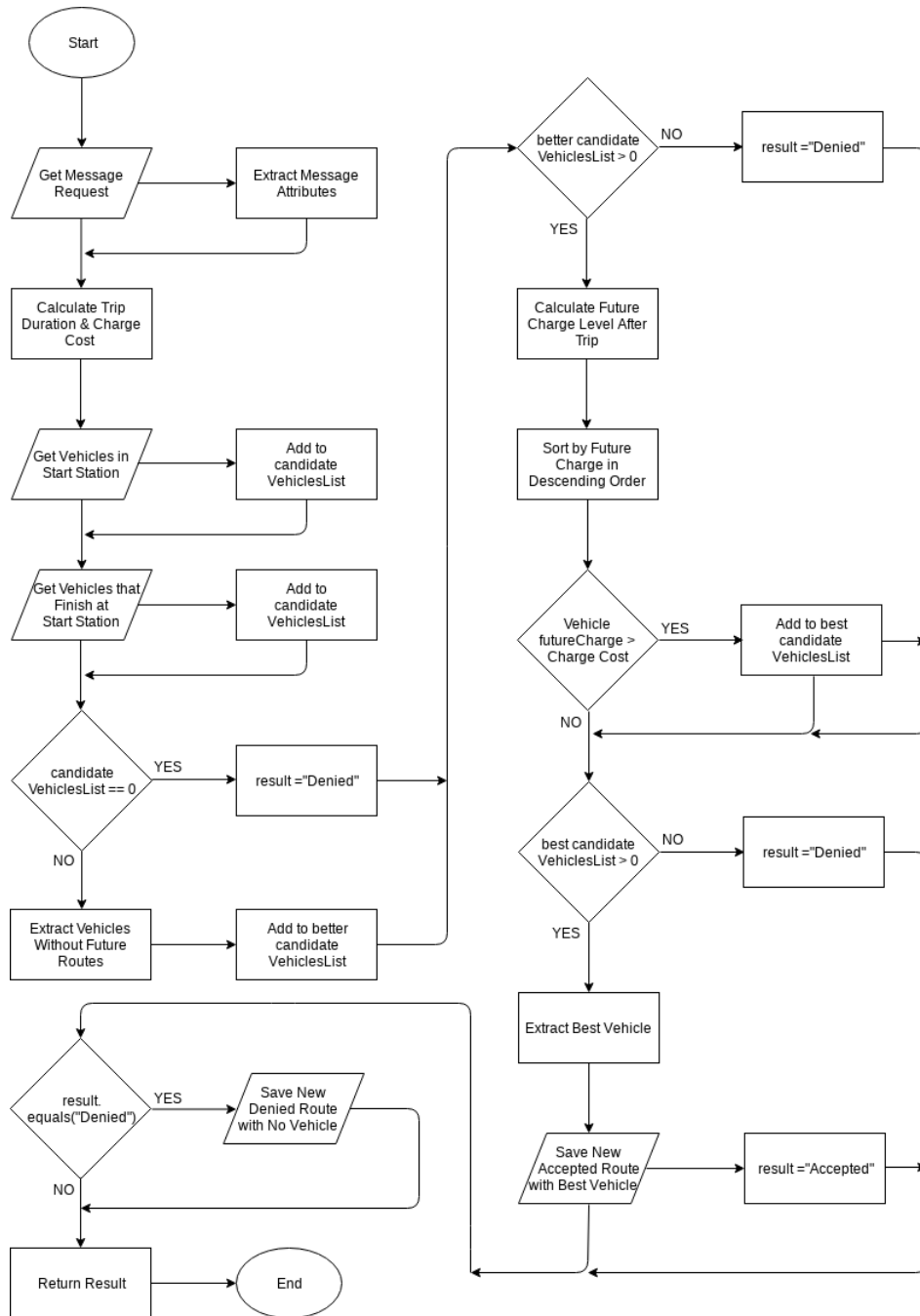


Figure 5.5.1: Short mode decision algorithm flowchart.

If there are no vehicles in the betterCandidateVehiclesList, the "result" string is set to "Denied", else we calculate and set the future charge for each vehicle by subtracting the charge cost from the current charge level of each vehicle. We sort the vehicles by future charge in descending order and if the future charge of a vehicle is bigger than the charge cost, we add it to the bestCandidateVehiclesList. Now, the bestCandidateVehiclesList contains all the vehicles that are suitable for the request trip. The first vehicle in the list has the maximum charge level, so it is the best vehicle. We assign it to the to the new route with the user request details, save the route to the database and set the "result" string to "Accepted". If there are no vehicles in the bestCandidateVehiclesList, the result is set to "Denied" and the route is saved as a denied route, without any vehicle details. The flowchart for Short mode algorithm is shown in Fig. 5.5.1.

In summary, the Short mode algorithm extracts the message requests attributes, gets the vehicles that currently are, or finish in the start station before start time and adds the vehicles without future routes to a list. We calculate the charge cost for the request trip, get the vehicles with enough charge and assign the best vehicle to the new route with the user request details. We save the route to the database and set the "result" string to "Accepted" (Table 5.5.1).

STEPS	SHORT MODE ALGORITHM
1	Extract message requests attributes
2	Get available vehicles in start station
3	Get available vehicles without future routes
4	Get available vehicles with enough charge
5	Assign best vehicle to new route
6	Save new route and result string

Table 5.5.1: Short mode algorithm overview and steps.

The Short mode algorithm has a relatively simple implementation, but its biggest drawback is that if a vehicle is assigned for a route that starts for example, after 6

hours, this vehicle is considered "locked/assigned" and can not be used for another route for those 6 hours. One way to tackle this problem is to restrict the minutes in the future that users are allowed to request a trip. For example, in the mobile application we can only allow requests for the next 60 minutes to be sent to the server, by editing the time control in the Request Screen.

5.6 LONG MODE DECISION ALGORITHM

A restriction that does not allow users to request vehicles at any time during the day, may not be a viable solution to the "locked/assigned" vehicles. In order to overcome this problem, we developed and implemented the Long mode algorithm that also considers the future routes of other vehicles and if they can substitute a current "locked/assigned" vehicle to make it free for use.

The Long mode algorithm gets the request message, extracts its attributes, and adds the vehicles that are currently in the start station and the vehicles that finish at the start station before the start time, to the candidateVehiclesList. Next, we get the vehicles without future routes and the vehicles with one future route and add them to betterCandidateVehiclesList and betterCandidateVehiclesListWithLateRoutes respectively. We sort the vehicles by future charge in descending order and if the future charge of a vehicle is bigger than the charge cost, we add it to the bestCandidateVehiclesList. Now, the bestCandidateVehiclesList contains all the vehicles that are suitable for the request trip. The first vehicle in the list has the maximum charge level, so it is the best vehicle. We assign it to the new route with the user request details, save the route to the database and set the "result" string to "Accepted". If there are no vehicles in the bestCandidateVehiclesList, the result is set to "Denied" and this concludes first part of the decision procedure that follows the logic of the Short mode algorithm.

At this point there are no vehicles without future routes and with enough charge level to cover the trip. So, we need to find a candidate vehicle with a future route, assign its route to a substitute vehicle and then assign this vehicle, which will not

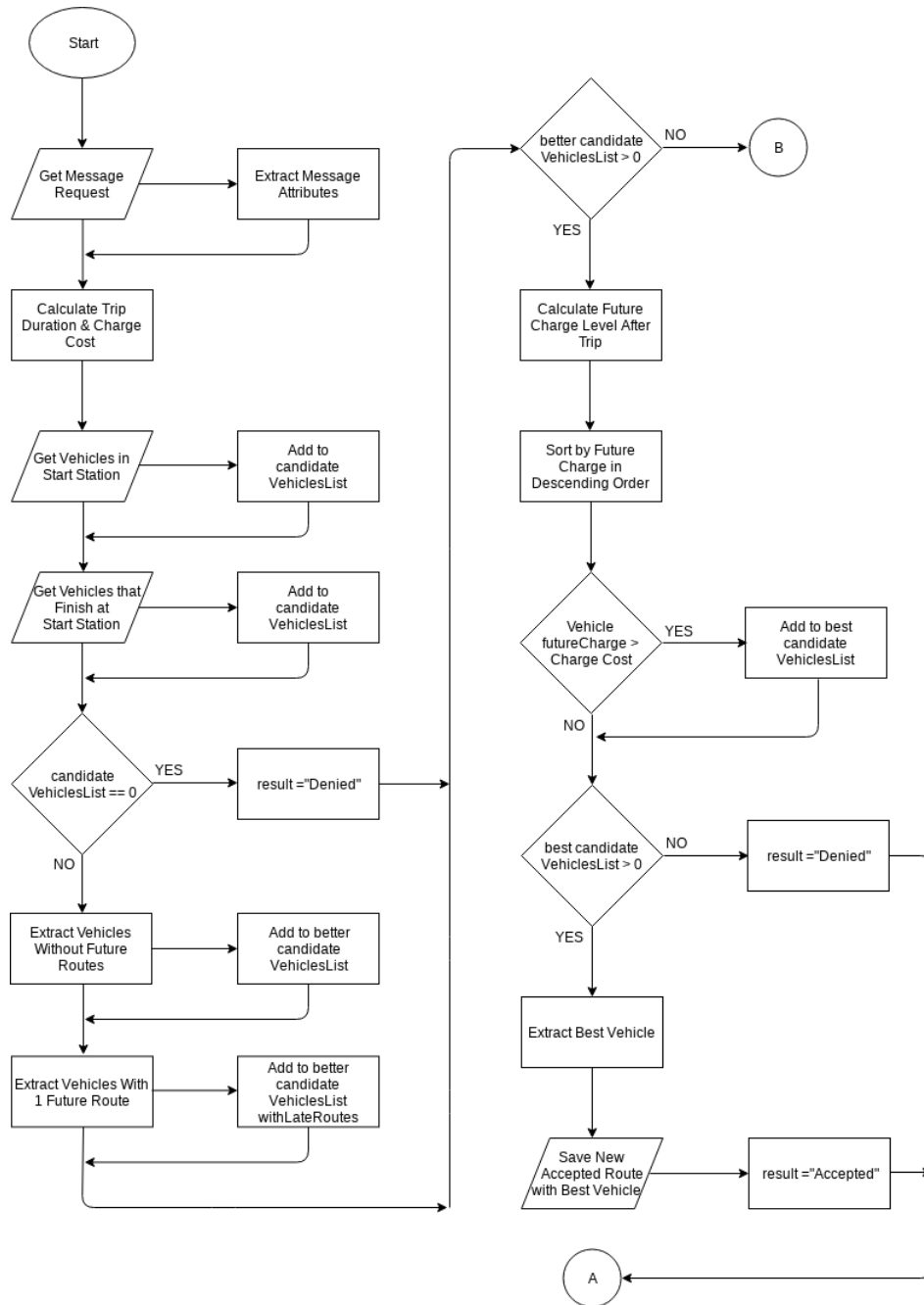


Figure 5.6.1: Long mode decision algorithm flowchart (part I).

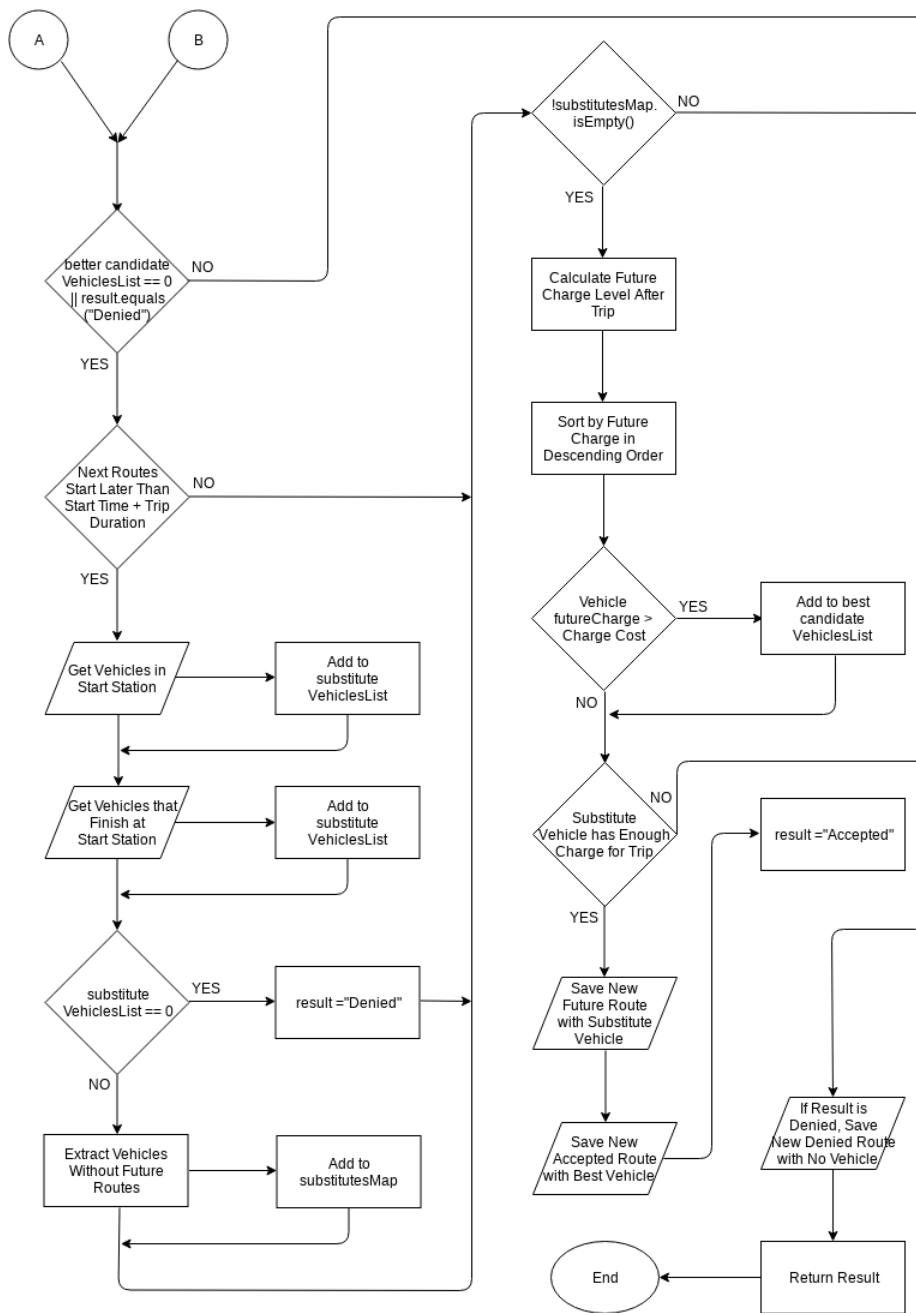


Figure 5.6.2: Long mode decision algorithm flowchart (part II).

have a future route now, to the user request trip. First, we check the future route of the vehicles in the `betterCandidateVehiclesListWithLateRoutes`. If this route starts later than the start time plus the trip duration of the user request, we need to find a substitute vehicle. We add the vehicles that are currently in the start station and the vehicles that finish at the start station before the start time of this future route, to the `substituteVehiclesList`.

Next, we add the substitute vehicles without future routes to the `substitutesMap`, that holds the better candidate vehicle ids and arraylists of substitute vehicles ids for those candidate vehicles. We sort them by future charge in descending order. If the substitute vehicle of the best candidate vehicle has enough charge for the future trip of the candidate vehicle, we assign the substitute vehicle to the candidate vehicle's future route and save the updated route to the database. Now, the best candidate vehicle is no longer "locked/assigned" and it is free for use. We assign it to the new route with the user request details, save the route to the database and set the "result" string to "Accepted". The flowchart for Long mode algorithm is shown in Fig. 5.6.1 and Fig. 5.6.2.

In summary, the Long mode algorithm gets the vehicles in the start station without future routes and those with one future route and adds them to lists. If there are vehicles with enough charge in the first list, we assign the best vehicle to the new route and save the route to the database, else we search for a substitute vehicle to replace the future route of a vehicle in the second list. We get substitute vehicles without future routes and with enough charge, and we assign the best substitute to the future route of the best vehicle in the second list. Now, the best vehicle in the second list is no longer "locked/assigned" and we assign it to the new route, save the route to the database and set the "result" string to "Accepted" (Table 5.6.1).

As we shall see later, we tested the two algorithms by generating a high number of requests. Although the Short mode is simpler in its implementation, the Long mode algorithm performed better in all cases with an average gain from 1.17% - 1.83% in the efficiency rate of the system. In general, the higher the number of

STEPS	LONG MODE ALGORITHM
1	Extract message requests attributes
2	Get available vehicles in start station
3	Get available vehicles without future routes
4	Get vehicles with one future route
5	Get available vehicles with enough charge
6	Assign best vehicle to new route
7	Save new route and result string
8	Else, get substitutes vehicles in start station
9	Get substitutes vehicles without future routes
10	Get substitutes vehicles with enough charge
11	Assign substitute vehicle to future route
12	Update future route with substitute vehicle
13	Assign best vehicle to new route
14	Save new route and result string

Table 5.6.1: Long mode algorithm overview and steps.

user requests, the better the efficiency is. This means that more requests get accepted and turned into actual routes and less requests are denied. A more detailed comparison between the two algorithms is available on the next chapter.

A car sharing scheme operating in a city with many requests per day, will benefit by converting more user requests to actual routes, thus increasing the customer satisfaction. And this is expected, as we designed the Long mode algorithm to also search for future routes of other vehicles that can substitute a current vehicle that already has a route assigned. This way, the vehicle can be regarded as free, ready to serve a new customer because its route is now assigned to the substitute vehicle.

6

Testing

6.1 TEST STRATEGY

Our test strategy includes three types of testing performed throughout the development procedure, using both black-box and white-box test design techniques. Black-box techniques include equivalence partitioning and boundary value analysis for proper input validation testing. In use case testing the system was evaluated when the most important user scenarios were executed. With white-box techniques, we measured the thoroughness of testing by creating test cases to achieve the highest percentage of statement and decision coverage.

Functional testing was performed in all phases of development, targeting all test levels from the smallest functional unit to the whole software system. During Phase V, we created a tool to generate routes with many different configurations and examined the results. Those randomly generated test cases were an excellent

way to assess the efficiency of the platform decision algorithms. In non-Functional testing we focused on security, performance and usability of the system. Alpha testing was also performed, where test scenarios were given to a small number of users in order to assess the system readiness and reveal possible problems in user experience. A summary of test cases is available in Appendix section 9.5.

6.2 FUNCTIONAL TESTING

The functional testing includes the testing and evaluation of the system functionalities and its components. Test conditions and test cases are derived from the requirements list and they can be divided into the following categories: navigation links, input validation, data consistency and interface testing. Navigation links functionality should always point to the specified web page, input validation includes checking the behavior of the system with empty, invalid and valid values. Creating, accessing, editing and deleting objects from the database ensures data consistency and interface testing verifies that communication is done properly.

Functional testing was performed in all levels of testing. The levels of testing are: component, integration, system and acceptance. In Phase I and Phase II where the main subsystems were developed, the testing of every component took place at each last step, before proceeding to the next phase. We made sure that the behaviour of the objects, classes and methods worked as expected before moving on to the next steps of development. In Phases III and IV we designed test cases for system integration testing. The platform scheduler and decision algorithms required that the interface communication between the two subsystems, the web platform and the mobile application, was working according to specifications. The web service and TCP connections tests were crucial, as these components are responsible for proper data exchange between the subsystems. Phase V included functional, non-functional, and acceptance testing in order to assess the readiness of the whole system, which we will discuss later during evaluation. An overview of functional testing with the test cases, their levels and phases are presented in Table 6.2.1.

TEST LEVEL	ACTIVITY/TEST CASE	PHASE
1. Component	Administrator login	I
2. Component	Manage administrators	I
3. Component	Manage users	I
4. Component	Manage stations	I
5. Component	Manage vehicles	I
6. Component	Manage routes	I
7. Acceptance	Web platform functionalities	I
8. Component	User login	II
9. Component	Manage user accounts	II
10. Component	Send vehicle requests	II
11. Component	Update existing user data	II
12. Component	Save request response to history	II
13. Acceptance	Mobile application functionalities	II
14. Integration	Access user accounts details	III & IV
15. Integration	Access stations details	III & IV
16. Integration	Send request to server	III & IV
17. Integration	Get response from server	III & IV
18. Component	Scheduler functionality	III
19. Acceptance	Scheduler functionality	III
20. Component	Short mode algorithm	IV
21. Component	Long mode algorithm	IV
22. Acceptance	Decision algorithms functionality	IV
23. System	Web platform functionalities	V
24. System	Mobile application functionalities	V
25. Acceptance	User acceptance testing	V

Table 6.2.1: Overview of functional testing with the test cases and their corresponding levels and phases.

In the last phase, to better evaluate the behavior of the system, we developed a tool to generate random routes based on different criteria. Our purpose was to also evaluate the platform decision algorithms, Short mode and Long mode, and how some different criteria could affect the efficiency rate. Those criteria are: the

maximum number of route requests in a day, the maximum amount of minutes that a user is allowed to request a vehicle in the future, and the start times density factor, that essentially means how "close" or how "further away" in time, the start times of the route requests are. We generated 100 test cases for 60 maximum requests per day and 100 cases for 120 maximum requests per day, with different configurations and assigned them to the Short mode and Long mode algorithms. We found that the Long mode algorithm performed better in all the tests, while the Short mode had very good performance, slightly behind Long mode, especially when used in test cases with 60 routes as the day limit.

6.3 NON-FUNCTIONAL TESTING

Non-functional is used to assess the quality characteristics of an application that can be quantified. Our non-functional testing includes three areas: security, performance and usability. In security testing we tried different scenarios to gain access to restricted content without the valid credentials and also tried to manipulate the system with different actions from the mobile application. Possible vulnerabilities of the web server setup were exposed with automated scanning from external websites like Tinfoil, which informed us about the overall security of the server and its defense against some very popular attacks, such as cross-site scripting and unencrypted password submissions.

The performance of the web platform was assessed with the automated test cases from our route generator tool and the very popular online tool from GTmetrix. With our tool, we generated the data for performing load testing with up to 120 simultaneous user requests. GTmetrix generated a report with the most important performance issues, including statistics about loading time, total page size and a waterfall chart with loading behavior. The mobile application overall performance was also tested both during development and during acceptance tests to evaluate the response time of the whole system.

TEST TYPE	ACTIVITY/TEST CASE	PHASE
1. Security	Try to bypass administrator login	V
2. Security	Try to bypass user login	V
3. Security	Web server online test	V
4. Performance	Load testing with route generator	V
5. Performance	Web server online test	V
6. Usability	Accessibility testing	V
7. Usability	Identity testing	V
8. Usability	Navigation testing	V
9. Usability	Content testing	V

Table 6.3.1: Overview of non-functional testing with the test cases and their corresponding types and phases.

In usability testing our objective was to identify usability problems related to accessibility, identity, navigation and content. With a series of heuristic rules, we evaluated how well the system is designed to be accessible by users with different browsers and devices. Identity in usability testing is about communicating the system purpose to the users with a consistent design. The navigation aspect of usability is to provide the user an easy way for navigating to different areas of the platform or the mobile application. The content fonts and sizes should also be consistent and descriptive for fast information locating and retrieving. Usability testing was performed primarily by us in the last phase of development. An overview of non-functional testing with the test cases, their types and phases are presented in Table 6.3.1.

6.4 USER ACCEPTANCE TESTING

User acceptance testing or alpha testing, is performed to identify all possible issues before releasing a product to the public. It is done by possible users of the product, while developers observe them and analyze their behavior. The tasks that need to be performed are a mix of the most important functional and non-functional requirements. It involves conducting a user testing with two groups of users. One

group performed acceptance testing as administrators on the web platform and the other group tested the mobile application functionalities.

The first step was to chose the most important functionalities of the system in order to create a list of tasks, from easier to harder, that the testers needed to perform. Then we documented other test specification details (Table 6.4.1), such as the number of testers, the number of tasks and the time allowed for the testers to complete them. During the testing procedure, we observed the behavior of the testers and took notes to see if they succeeded or failed at the tasks given. When they finished, we conducted a small interview to learn more about how they felt at specific occasions, how they reacted and why.

This procedure of testing our system gave us valuable insights about areas that might need improvements, or weaknesses that have not been recognized from previous testing activities. Ideally, after that process we would be ready to perform open beta testing, allowing a bigger number of users to evaluate our software solution.

CHARACTERISTICS	SPECIFICATIONS
Type of testers	Inexperienced and experienced
Number of groups	2 (web platform + mobile app)
Number of people per group	5
How the tasks are revealed	One at a time, from easier to difficult
Maximum time allowed	30 minutes
Users can leave anytime	Yes
Questions allowed during the test	No

Table 6.4.1: Specifications for alpha testing.

6.5 DECISION ALGORITHMS RESULTS

With the route generator tool, we had the opportunity to see how the decision algorithms produce different efficiency rate results. With 60 maximum requests per day, we experimented with 60 - 600 minutes between the requests and different values of the requests allocation density within a day. This factor shows how "concentrated" or close, the start times of the requests are, and we tested with 5 different density factors.

First, we tested with 60 maximum requests per day. The Short mode produced an efficiency rate (the percentage of the requests that get accepted by the system) from 41.00% - 54.33% and the Long mode from 42.33% - 55.33% and it performed better with an average gain of 1.17%. When testing with 120 maximum requests per day, the Short mode produced efficiency rate from 27.67% - 45.50% and the Long mode from 27.83% - 47.83% and it performed better with an average gain of 1.83%.

Looking closer at the charts in Fig. 6.5.1 and Fig. 6.5.2, we can draw some interesting observations: (1) the Long mode algorithm performs better than the Short mode in all tests, (2) the Long mode offers more gain in the efficiency of the system when the maximum number of requests increase, (3) when the maximum time between the start times of the requests increases, the efficiency increases, (4) the overall efficiency seems not to be influenced by the requests allocation density within the day, (5) when the requests within the day increase, the efficiency of the system drops (for more detailed results see Appendix section 9.6)

The above observations are not conclusive because we have not tested extensively with the generator tool (200 test cases are not enough). But we can take them into account and say that in most cases, and especially when we have a higher number of requests, the Long mode algorithm is preferred and should be used.

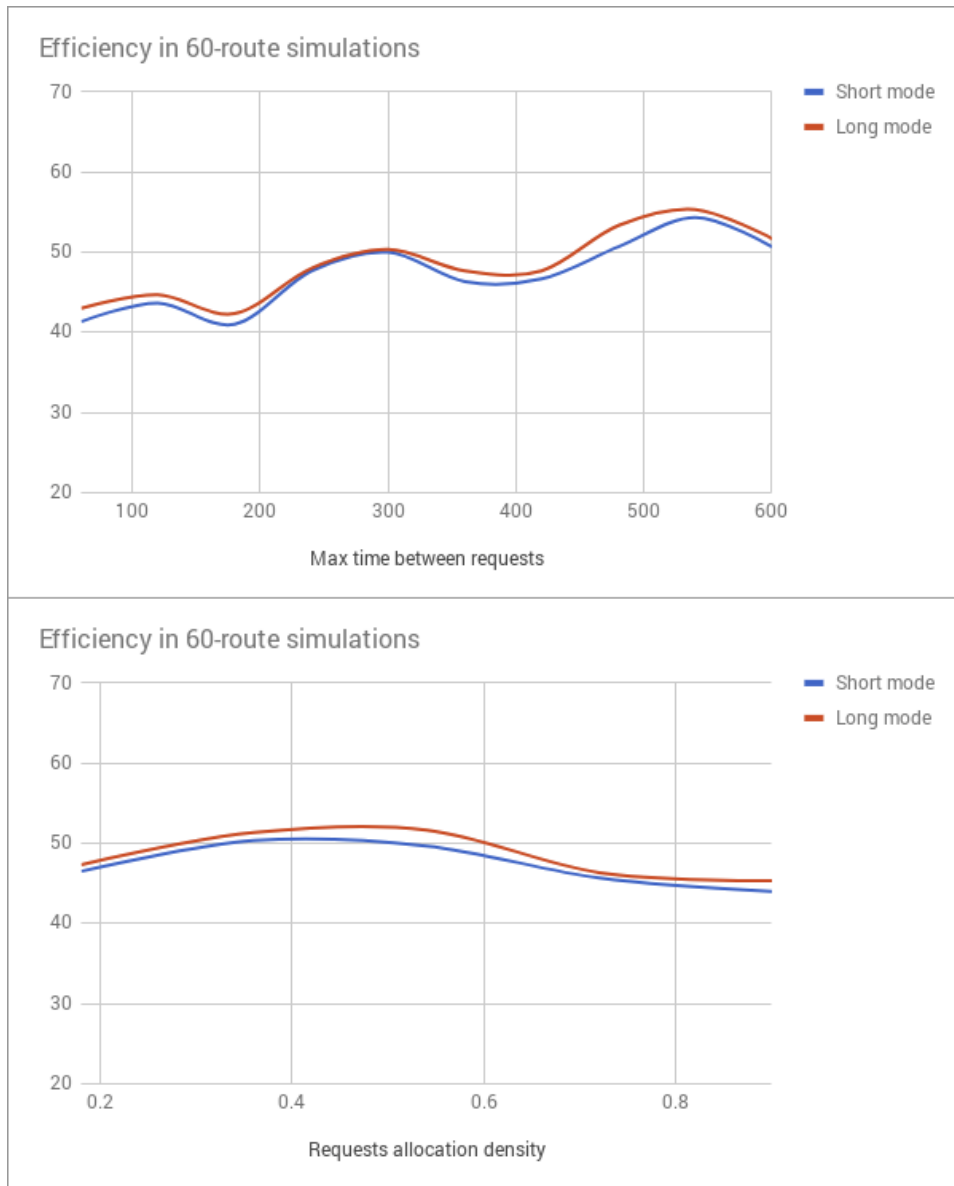


Figure 6.5.1: Efficiency rate graphs when the system accepts 60 requests for routes. The graphs show the efficiency related to the maximum time between requests and the requests allocation density.

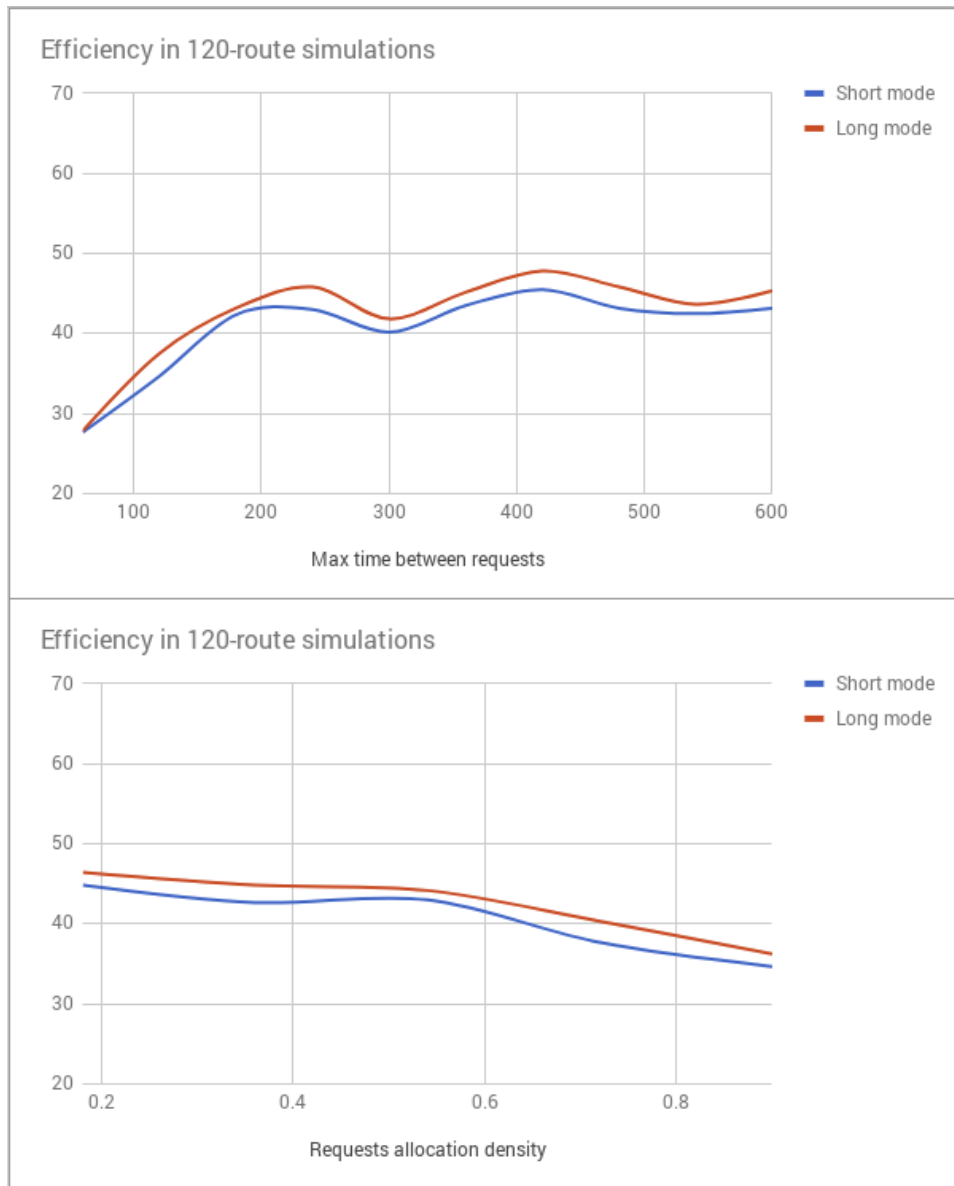


Figure 6.5.2: Efficiency rate graphs when the system accepts 120 requests for routes. The graphs show the efficiency related to the maximum time between requests and the requests allocation density.

6.6 EVALUATION

The functional testing throughout the development lifecycle revealed a few bugs that were mostly easy to fix. All test cases in component, integration, system and acceptance level were characterized as "Passed". The web platform works as expected, managing administrators, users, stations vehicles and routes. The mobile application functionalities exchanged the correct information with the platform and the database. The areas with the most risk were the platform scheduler and the decision algorithms. Exploratory testing exposed some minor bugs, which were also fixed.

Non-functional testing revealed that the security implemented, protects the administrator and user accounts from unauthorized access. Spring Security successfully filters out malicious requests and session variables allow only logged in administrators to browse the web platform. The communication between the web platform and the mobile application is realized via a RESTful web service, protected by an OAuth2 framework that uses expiring tokens, so accessing user accounts and station details is very secure. Vehicle requests from the mobile applications are sent as encrypted messages via a TCP connection and even if someone could steal the traffic from the communication medium, the information from the encrypted messages would be useless. Web server assessment did not reveal any security concerns with the way the server was setup and configured.

Performance testing was done in two modes: a continuous mode where the system was processing simulation XML files for a period of 30 minutes, and a standby mode where the service was running, waiting for incoming requests. Looking at the monitoring graphs we can see that the server CPU usage was in the range of 6.50% - 10.15% when the system was processing simultaneous requests. Memory usage increased from 59.70% to 65.85% and local disk usage reached 17.79%. In standby mode, CPU usage dropped to 1.66% (Fig. 6.6.1). GTmetrix showed a web page load time of 1.3 seconds and a total page size of 325KB. The performance of the web platform was great and we manually assessed the mobile application speed



Figure 6.6.1: Server monitoring graphs during continuous and standby web platform operation modes.

and responsiveness and we were completely satisfied.

In usability testing, we assessed the accessibility, identity, navigation and content. All functionalities are accessible from different browsers and devices. The layout of the web platform is optimized for full HD widescreen monitors. The mobile application was tested on different phones and proved very easy to use and make requests to the server. Alpha testing evaluated the functionalities and usability of the whole system. Users accomplished all tasks without any difficulties. Especially mobile users, expressed that the mobile application was very easy with a nice user interface.

Taking all tests into account, we can say that our software solution is thoroughly tested and achieves its aim and objectives. The web platform is easy to install and configure and a car sharing company can manage its assets, like stations, vehicles and routes, easily. The mobile application allows users to select a start station, their destination, send their request for a vehicle and receive the response from the server in a fast, reliable and secure way.

The overall evaluation of the software system is almost excellent, fulfilling all the requirements of the project. We have produced a backend and frontend sys-

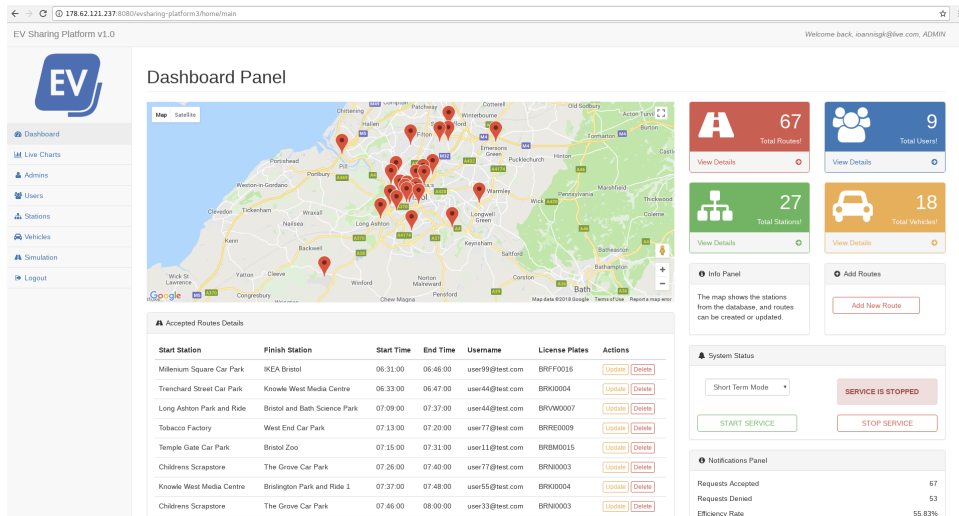


Figure 6.6.2: Screenshot of the Dashboard page in the web platform.

tem that only needs beta testing in order to be ready for production and serve car sharing companies with EVs. From the mockup designs to implementing the user interface and every functionality, we offer a software solution with a great user experience. Fig. 6.6.2, 6.6.3 and 6.6.4 show screenshots of the Dashboard page, the Manage Stations page and the Live Charts page in the web platform. Fig. 6.6.5 and Fig. 6.6.6 show screenshots of My Profile, User History, Open Map screens and the navigation menu in the mobile application.

The deliverables of the project are freely available in the following links:

- source code of the web application project file:
<https://github.com/ioannisgk/evsharing-platform3-release>
- source code of the Android application project file:
<https://github.com/ioannisgk/evsharing-app-release>
- video demonstration: <https://youtu.be/flyixErIE-A>

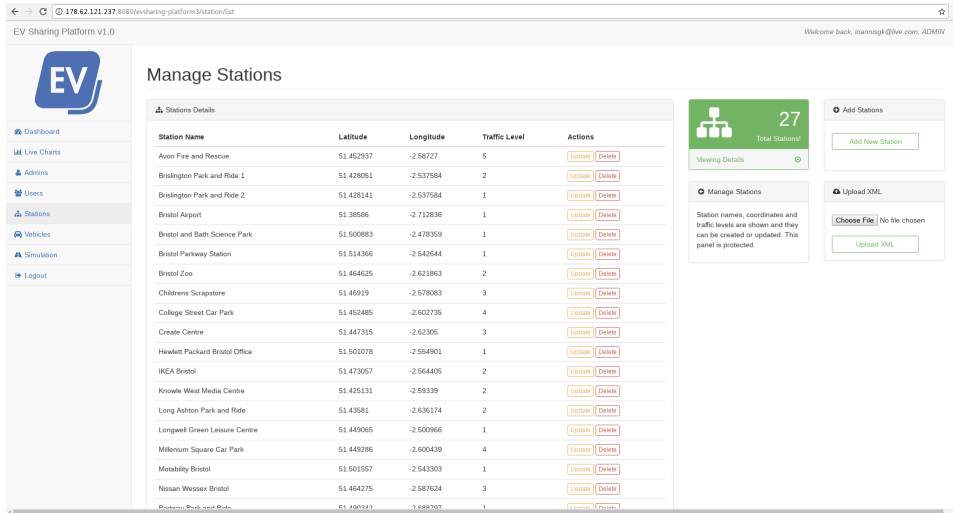


Figure 6.6.3: Screenshot of the Manage Stations page in the web platform.

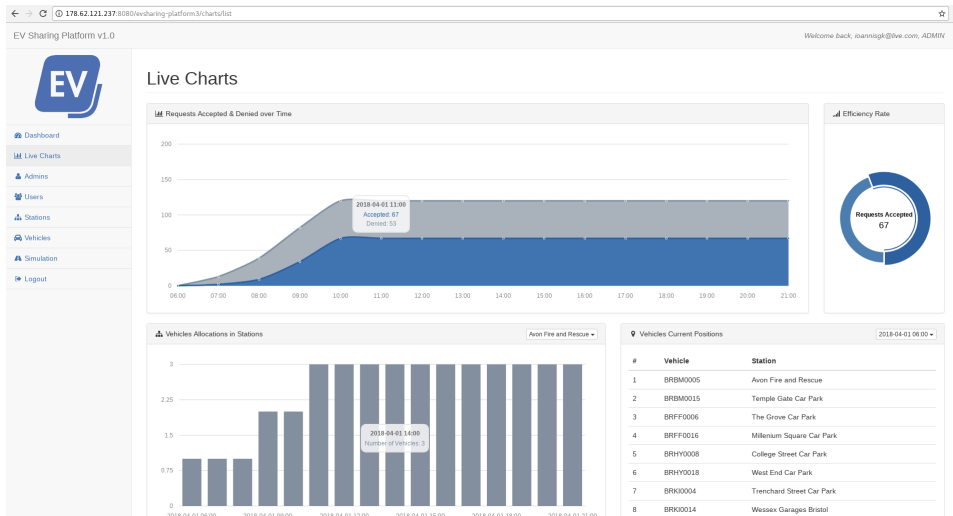


Figure 6.6.4: Screenshot of the Live Charts page in the web platform.

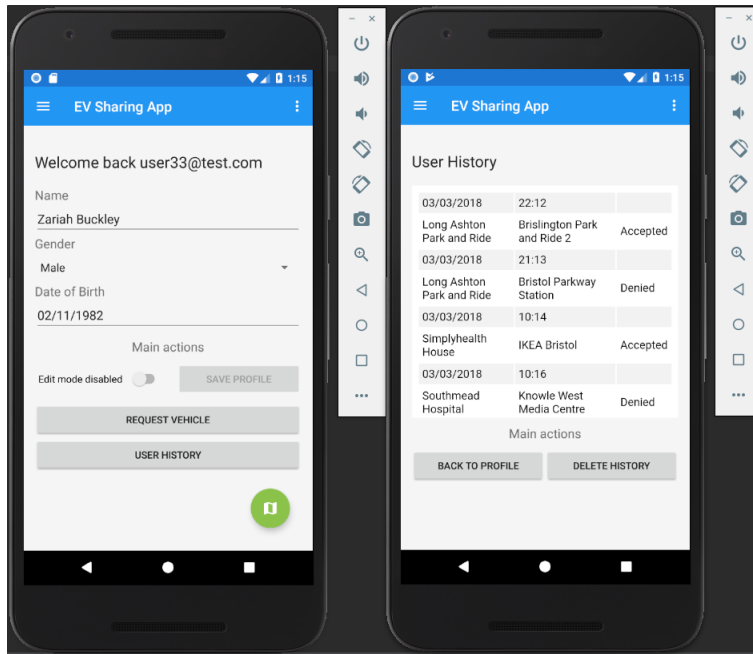


Figure 6.6.5: Screenshots of My Profile and User History screens.

6.7 LIMITATIONS

There are some limitations identified in the server configuration level and in the overall system level, as an application intended to consumers. During testing, we noticed a couple of times that the server socket could not be bound to the web platform. This usually happens when using a VPS and not a dedicated server, because a VPS shares its resources with other virtual machines and applications. This issue happens rarely and a workaround is to stop Tomcat service on the server and restart it. It is a very easy procedure but in our server it takes around 20 minutes for Tomcat to restart and reload the web platform. A dedicated server would be the best solution for a car sharing company.

In the system level, the web platform serves users that request a vehicle during a given time period, which is a day. Users can not make requests for a specific date, e.g. one or two days in the future. This is mainly due to the special needs of EVs: they have to be fully recharged and to be relocated to the optimal stations so

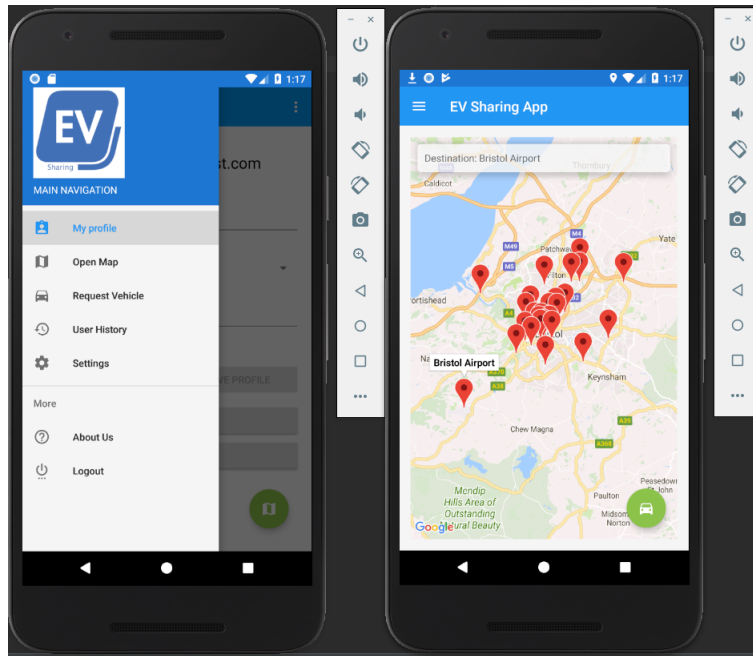


Figure 6.6.6: Screenshots showing the menu and the Open Map screen.

that the car sharing fleet will be ready for the next day of operation. The mobile application allows users to select only a start station, a finish station and a time within the day of operation and this is an important disadvantage.

Another limitation is that the mobile application is intended for users with Android devices only. There is a significant market share with iOS phones that could benefit from the service. A car sharing company should not just ignore those users. Urban mobility upgrade depends on all of us and we should all be a part of it.

7

Conclusions

7.1 SUMMARY

Urban mobility has been essential for the well being of its residents, as transport is necessary to their everyday lives. So, cities started to get more crowded and that led to increased motorization. As a result, too many vehicles in urban areas created a burden that is bound to a city and becomes heavier and heavier every day. Air pollution, noise, the ecological footprint, congestion, road accidents and insufficient transport are just some of the negative consequences. Our main medium for travelling is the ICE vehicle, which uses fossil fuels and it is energy ineffective.

Cities need a new kind of automobile. A vehicle that uses renewable energy can help greatly in their sustainable development. EVs, and in particular BEVs with no harmful emissions and a high level of efficiency, seem like an excellent solution. They can benefit from connectivity and wireless networks, and offer real time traf-

fic information, remote diagnostics and crash notifications. However, there are some challenges that we need to overcome for a wider adoption of BEVs, like the battery cost, the charging time and the charging infrastructure. Worldwide initiatives help the EV industry, and governments with OEMs pushed the number of EVs on the roads to about 1.26 million (Global EV Outlook 2016).

Car sharing is an alternative personal mobility system for urban environments. With one-way, two-way or free floating car sharing schemes, the costs of the vehicle is divided to a large group of people, making it the most affordable option for low income inhabitants. Shared vehicle ownership is the most important advantage of car sharing as every shared car removes 4 to 23 cars from the roads (Vairani, 2009). This means decreased GHG emissions, traffic, accidents, more parking and a great efficiency in vehicle use. Although car sharing faces important challenges like the equal distribution of the vehicles at all times and location privacy, it is considered as an excellent alternative that protects the environment and diminishes all the negative effects of traffic and congestion.

From all those findings, the necessary ingredients for our mobility upgrade are obvious. We need EVs that use energy from renewable sources, to be integrated with car sharing systems. BEVs are the most suitable for this job and car sharing companies are in a position to give EV technology the boost it needs so much. Zipcar and other companies can achieve this goal and that will release positive impacts to our personal mobility, the environment and the economy of our cities. The convergence of EVs and car sharing systems will help us build better, sustainable cities with better and more reliable personal mobility. Bringing two "green" ideas together, the use of EVs and the car sharing systems, concludes the first part of our proposed urban mobility upgrade.

The second part of our proposed urban mobility upgrade is a software solution that will enable businesses to manage the use of EVs in one-way car sharing services. We designed, implemented and tested a system that consists of an online web platform and a mobile application. The web platform allows administrators to manage users, stations, vehicles, routes and accept or deny vehicle requests. The mobile application lets users register, login, see the available stations

in their area and send vehicle requests for trips to the web platform. The platform executes an algorithm in order to decide whether to accept or deny incoming requests, ensuring that the highest number of users will be served within a day. For this purpose we developed two algorithms, the Short mode and the Long mode, where the system searches for free vehicles or vehicles that can substitute current "locked/assigned" ones so that it can serve the maximum number of users.

The platform scheduler gathers the required data from the database and predicts the number of available vehicles in a specific station, the current station of a specific vehicle and its current charge level. All predicted data are updated every 10 seconds and displayed in the Live Charts page with diagrams and charts. The simulation functionality on the web platform allowed us to test multiple generated requests at once and examine the behavior of the system when using the Short mode or the Long mode algorithms. Results showed that Long mode performed better in all tests, especially when we had a higher number of requests within a day. It presented an average gain of 1.83% in efficiency rate in 120 requests per day when compared to the Short mode.

Our software solution successfully fulfills its aim and objectives. Carefully designed algorithms process user requests, ensuring that the highest number of users are served within a given time period. Both the web platform and the mobile application were thoroughly tested, offering a system that is:

- Reliable, easy to use by administrators and users, with great user experience
- Safe and secure, by protecting user accounts with OAuth2 framework and encrypting messages between the platform and the mobile application
- Easy to install and configure with the use of XML files for maps & vehicles
- Universal, allowing effortless adoption by car sharing companies with EVs
- Maintainable and extensible, since it is developed with Spring MVC, a well known open source development framework

A virtual demo (screenshots of various functionalities) and a presentation can be found in Appendix sections 9.7 and 9.8.

7.2 FUTURE WORK

The future work for this project should be focused on optimizing the platform decision algorithms. We have tested the Short mode and the Long mode algorithms but there are many options and room for experimentation ahead. Should we use only one algorithm? Should we develop an "engine" to detect specific parameters and apply the best algorithm for each situation? There are many possibilities to increase the system efficiency and serve more users.

The algorithms accept as input the user request parameters, process data and decide if a request should be characterized as "Accepted" or "Denied". We can measure the performance of the algorithm by checking its efficiency rate. With a vast search field as input, Evolutionary Algorithms seem like a very good choice for improving our system. We can develop a system that creates a population of solutions (parameters to our decision algorithms), evolve it and generate new solutions that can be tested and evaluated. Applying Evolutionary Algorithms to optimize the effectiveness of our system, is one of the recommended paths for future work.

Moving closer to the field of Artificial Intelligence, we can benefit from Machine Learning techniques and Neural Networks. We can improve the performance of existing algorithms that learn from experience and previous encounters with the input data. Using our tool generator to create a large pool of user requests, will give us the opportunity to train our algorithms with this data and make them better over time. We can even use Genetic Algorithms to generate the optimal settings to train a Neural Network and generate new sets of feature selections that will help greatly in further increasing the efficiency rate of the system.

Concluding, the future work should be focused on how Evolutionary Algorithms and Machine Learning can improve the efficiency of our software solution. We live in a new era of technological evolution and a car sharing company with EVs can offer great value to the sustainable development of the cities. With our proposed software solution, the urban mobility upgrade will make our every day lives in the cities healthier, eco-friendly and pleasurable.

8

References

Banister, D. (2005). *Unsustainable Transport: City transport in the new century*. London: Taylor & Francis.

Bardhi, F. and Eckhardt, G. (2012). Access-Based Consumption: The Case of Car Sharing. *Journal of Consumer Research*, 39(4), pp.881-898.

Botsford, C., Szczepanek, A. (2009). Fast charging vs. slow charging: Pros and cons for the new age of electric vehicles. In *International Battery Hybrid Fuel Cell Electric Vehicle Symposium*.

Brennan, J. and Barder, T. (2016). *Battery Electric Vehicles vs. Internal Combustion Engine Vehicles*. Arthur D. Little.

Buchanan, C. and Gunn, S. (2015). *Traffic in Towns: A study of the long term problems of traffic in urban areas*. London: Routledge.

Burke, A. (2007). Batteries and Ultracapacitors for Electric, Hybrid, and Fuel Cell Vehicles. *Proceedings of the IEEE*, 95(4), pp.806-820.

Burns, L. D., Mc, J. B., Jablonski, N. G., Chaplin, G., Casagrande, R., and Voldman, S. H. (2002). 64 Vehicle of Change. *Scientific American*, 287(4), pp.64-73.

Chan, C. (2002). The State Of The Art Of Electric And Hybrid Vehicles. *Proceedings of the IEEE*, 90(2), pp.247-275.

Clement-Nyns, K., Haesen, E. and Driesen, J. (2010). The Impact of Charging Plug-In Hybrid Electric Vehicles on a Residential Distribution Grid. *IEEE Transactions on Power Systems*, 25(1), pp.371-380.

Cox, A. (2001). BEST Experience from Bristol. *Town and Country Planning*, 70(5), pp.145.

Dinger, A., Martin, R., Mosquet, X., Rabl, M., Rizoulis, D., Russo, M., and Sticher, G. (2010). Batteries for electric cars: Challenges, opportunities, and the outlook to 2020. *The Boston Consulting Group*, 7, 2017.

Downs, A. (2004). *Still Stuck in Traffic: Coping with peak-hour traffic congestion*. Washington, D.C.: Brookings Institution Press.

Droege, P. (2008). *Urban Energy Transition: From Fossil Fuels to Renewable Power*. Oxford: Elsevier.

European Environment Agency (2016). *Electric Vehicles in Europe*. Luxembourg: Publications Office of the European Union.

Fellows, N. and Pitfield, D. (2000). An economic and operational evaluation of urban car-sharing. *Transportation Research Part D: Transport and Environment*, 5(1), pp.1-10.

Gaggi, S., Fluhrer, T., and Janitzek, T. (2013). *Innovation In Urban Mobility: Policy Making And Planning*. Luxembourg: Transport Research and Innovation Portal.

Garcia-Valle, R. and Peças Lopes, J. (2013). *Electric Vehicle Integration into Modern Power Networks*. New York, NY: Springer Science & Business Media.

Global EV Outlook (2016). *Beyond One Million Electric Cars*. Paris: International Energy Agency.

Gwilliam, K. (2002). *Cities on the move: a World Bank urban transport strategy review*. Washington, DC: World Bank.

Helmers, E. and Marx, P. (2012). Electric Cars: Technical Characteristics and Environmental Impacts. *Environmental Sciences Europe*, 24(1), p.14.

Herzog, A. V., Lipman, T. E., and Kammen, D. M. (2001). Renewable Energy: A Viable Choice. *Encyclopedia of Life Support Systems (EOLSS)*. Forerunner Volume- 'Perspectives and Overview of Life Support Systems and Sustainable Development.

Holden, E. (2007). *Achieving Sustainable Mobility: Everyday and leisure-time travel in the EU*. Ashgate Publishing, Ltd.

Hopwood, B., Mellor, M. and O'Brien, G. (2005). Sustainable Development: Mapping Different Approaches. *Sustainable Development*, 13(1), pp.38-52.

Jenks, M. and Dempsey, N. (2005). *Future Forms and Design for Sustainable Cities*.

London: Routledge.

Katzev, R. (2003). Car Sharing: A New Approach to Urban Transportation Problems. *Analyses of Social Issues and Public Policy*, 3(1), pp.65-86.

Kennedy, C., Miller, E., Shalaby, A., Maclean, H. and Coleman, J. (2005). The Four Pillars of Sustainable Urban Transportation. *Transport Reviews*, 25(4), pp.393-414.

King, J. (2007). *The King Review of Low-Carbon Cars Part I: The potential for CO₂ reduction*. Crown.

King, J. (2008). *The King Review of Low-Carbon Cars Part II: Recommendations for action*. Crown.

Laarabi, M. H., and Bruno, R. (2016). A Generic Software Framework for Carsharing Modelling Based on a Large-Scale Multi-agent Traffic Simulation Platform. In *ABMUS@ AAMAS*, pp.88-111.

Lisserre, M., Sauter, T. and Hung, J. (2010). Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics. *IEEE Industrial Electronics Magazine*, 4(1), pp.18-37.

McKay, D. (2008). *Sustainable Energy: Without the hot air*. UIT Cambridge Ltd.

McKinsey (2014). *Electric Vehicles In Europe: Gearing up for a new phase?* Amsterdam: Amsterdam Roundtable Foundation.

Milgram, S. (1970). The Experience of Living in Cities. *Science*, 167(3924), pp.1461-1468.

Millard-Ball, A., Murray, G., Schure, J., Fox, C. and Burkhardt, J. (2005). *Car-sharing: Where and how it succeeds* (Vol. 108). Washington, D.C: Transportation Research Board.

Mitchell, W., Borroni-Bird, C. and Burns, L. (2010). *Reinventing the Automobile*. Cambridge, Mass.: Massachusetts Institute of Technology.

Popa, R. A., Balakrishnan, H., and Blumberg, A. J. (2009). *VPriv: Protecting privacy in location-based vehicular services*.

Rigas, E., Ramchurn, S. and Bassiliades, N. (2015). Algorithms for Electric Vehicle Scheduling in Mobility-on-Demand Schemes. *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp.1339-1344.

Rodrigue, J., Comtois, C. and Slack, B. (2006). *The Geography of Transport Systems*. London: Routledge.

Schewe, P. F. (2007). *The Grid: A Journey Through The Heart Of Our Electrified World*. National Academies Press.

Schrank, D. and Lomax, T. (2007). *The 2007 Urban Mobility Report*. [College Park, Tex.]: Texas Transportation Institute, Texas A & M University.

Schrank, D., Eisele B. and Lomax, T. (2012). *TTI's 2012 Urban Mobility Report*. [College Park, Tex.]: Texas Transportation Institute, Texas A & M University.

Shaheen, S. and Cohen, A. (2007). Growth in Worldwide Carsharing: An International Comparison. *Transportation Research Record: Journal of the Transportation Research Board*, 1992, pp.81-89.

Sperling, D. and Gordon, D. (2009). *Two Billion Cars: Driving toward sustainabil-*

ity. Oxford: Oxford University Press.

Suzuki, H., Cervero, R. and Iuchi, K. (2013). *Transforming Cities with Transit: Transit and land-use integration for sustainable urban development*. Washington, DC: World Bank Publications.

Tietge, U., Mock, P., Lutsey, N. and Campestrini, A. (2016). Comparison of leading electric vehicle policy and deployment in Europe. *International Council on Clean Transportation*.

Turrentine, T., McCarthy, R., Nesbitt, K., Cunningham, J., and Boone, J. (2010). Taking Charge: Establishing California Leadership in the Plug-In Electric Vehicle Marketplace. *California Electric Plug-In Vehicle Collaborative*, 21.

Vairani, F. (2009). *BitCar: Design Concept for a Collapsible Stackable City Car*. Massachusetts: Massachusetts Institute of Technology.

Van Audenhove, F., Pourbaix, J., Dauby, L. and Korniiichuk, O. (2014). *The Future of Urban Mobility 2.0: Imperatives to Shape Extended Mobility Ecosystems of Tomorrow*. Arthur D. Little and UITP 2014.

Volkswagen Academy (2013). *Basics of Electric Vehicles*. Volkswagen Group.

Weikl, S. and Bogenberger, K. (2013). Relocation Strategies and Algorithms for Free-Floating Car Sharing Systems. *IEEE Intelligent Transportation Systems Magazine*, 5(4), pp.100-111.

9

Appendices

9.1 GANTT CHART

The project Gantt Chart shows the various activities and tasks over time (Fig. 9.1.1).

The main phases of development are the following:

- Phase I - Web platform
- Phase II - Mobile application
- Phase III - Platform scheduler algorithm
- Phase IV - Platform decision algorithms
- Phase V - Testing and evaluation

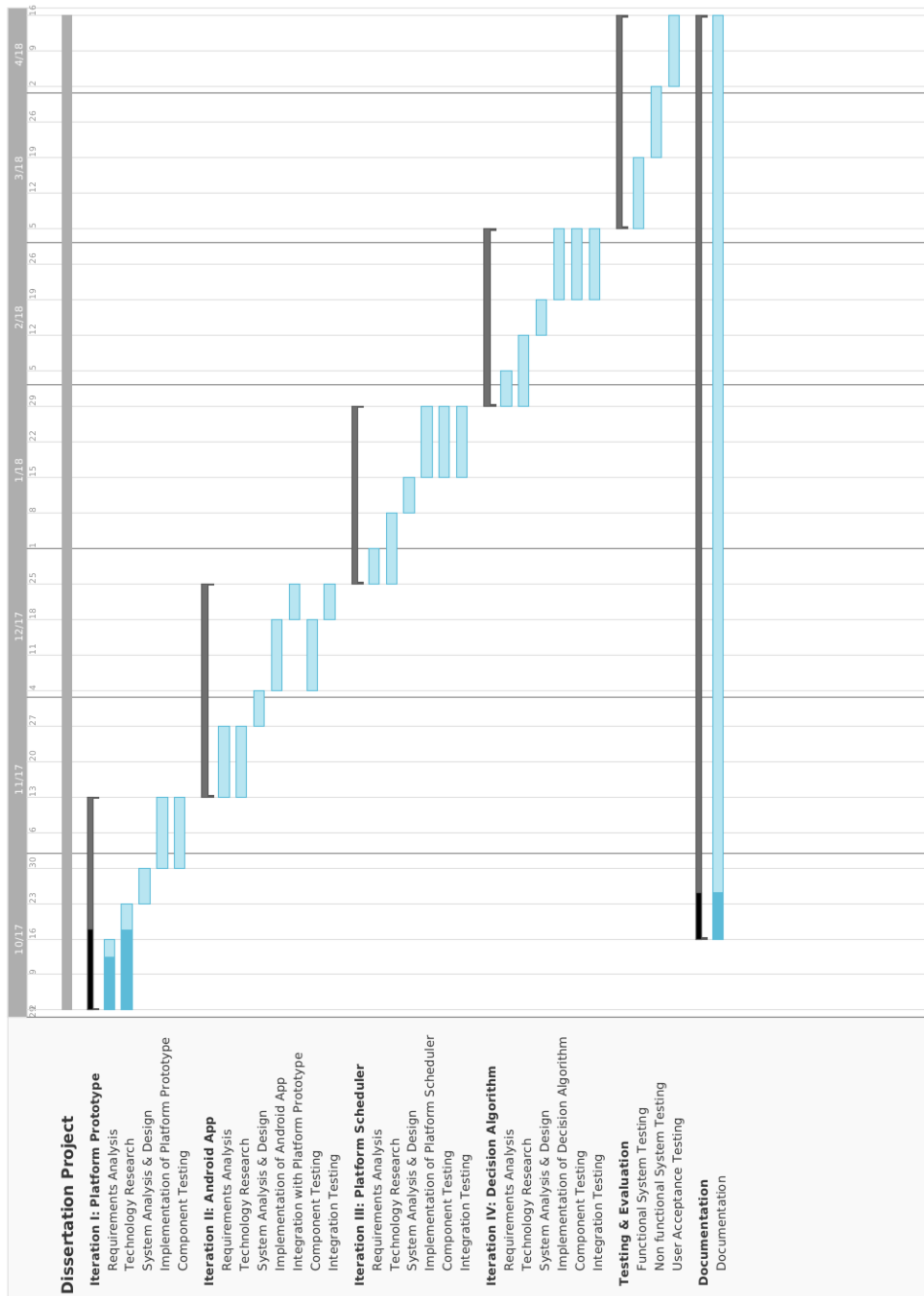


Figure 9.1.1: Project Gantt Chart.

9.2 PROJECT PLAN

Tables 9.2.1 - 9.2.4 show the tasks allocations during all phases of development. The dependent task is also shown along with the estimated time in weeks. The proposed milestone dates are the following: Phase I - 10 Nov 2017, Phase II - 22 Dec 2017, Phase III - 26 Jan 2018, Phase IV - 2 March 2018, Phase V - 13 April 2018.

TASK	PHASE I - WEB PLATFORM	DEPENDS ON	WEEKS
A1	Requirements analysis	-	2
A2	Technology research	-	3
A3	System analysis & design	A1, A2	1
A4	Implementation of platform	A3	2
A5	Component testing (milestone)	concurrent A4	2

Table 9.2.1: Tasks allocation for Phase I. Milestone date 10 Nov 2017.

TASK	PHASE II - MOBILE APP	DEPENDS ON	WEEKS
B1	Requirements analysis	-	2
B2	Technology research	-	2
B3	System analysis & design	B1, B2	1
B4	Implementation of Android app	B3	2
B5	Integration with platform	A5, B4	1
B6	Component testing	concurrent B4	2
B7	Integration testing (milestone)	concurrent B5	1

Table 9.2.2: Tasks allocation for Phase II. Milestone date 22 Dec 2017.

TASK	PHASE III - SCHEDULER	DEPENDS ON	WEEKS
C1	Requirements analysis	A5	1
C2	Technology research	A5	2
C3	System analysis & design	C1, C2	1
C4	Implementation of scheduler	B7, C3	2
C5	Component testing	concurrent C4	2
C6	Integration testing (milestone)	concurrent C4	2

Table 9.2.3: Tasks allocation for Phase III. Milestone date 26 Jan 2018.

TASK	PHASE IV - DECISION	DEPENDS ON	WEEKS
D1	Requirements analysis	C6	1
D2	Technology research	C6	2
D3	System analysis & design	D1, D2	1
D4	Implementation of decision	C6, D3	2
D5	Component testing	concurrent D4	2
D6	Integration testing (milestone)	concurrent D4	2

Table 9.2.4: Tasks allocation for Phase IV. Milestone date 2 March 2018.

Phase V includes three tasks: functional testing, non-functional testing and user acceptance testing, where we allocate 2 weeks per task in order to have enough time to fix possible bugs.

9.3 WEB PLATFORM CLASS DIAGRAMS

The web platform class diagrams are below:

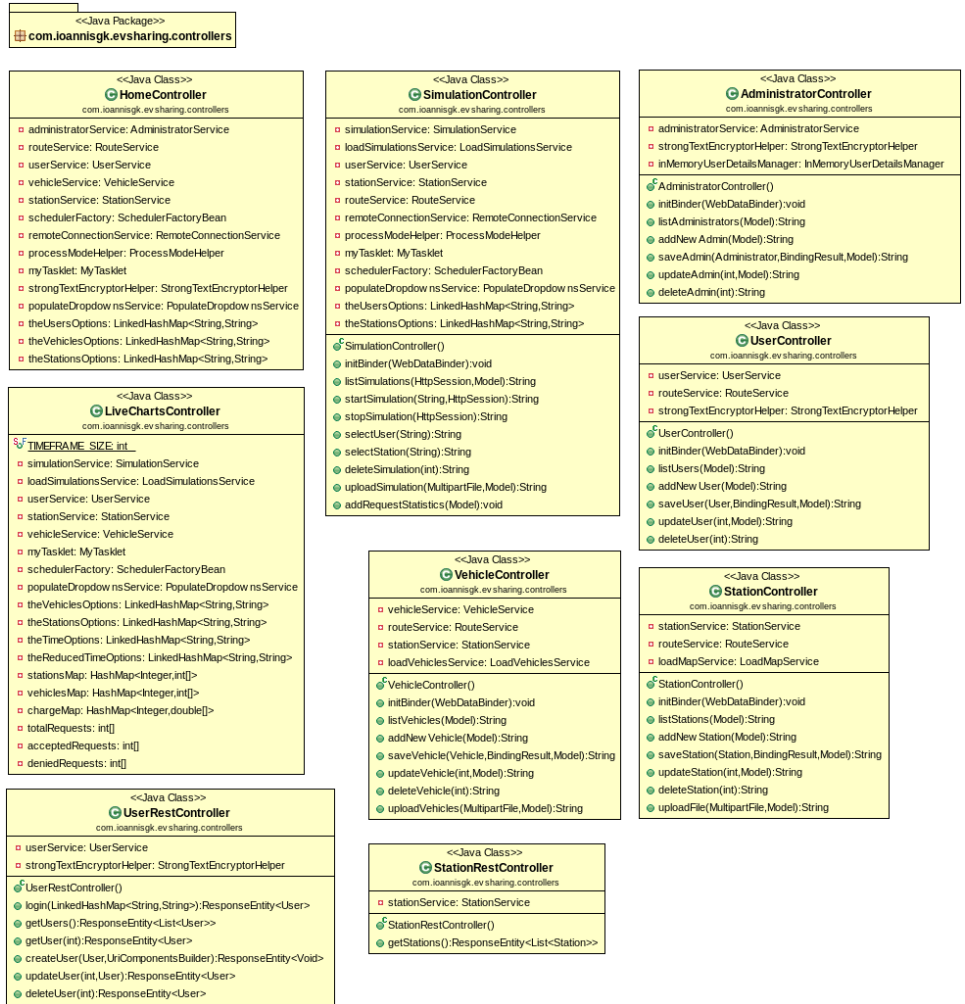


Figure 9.3.1: Class diagram for Controllers package.



Figure 9.3.2: Class diagram for Entities package.

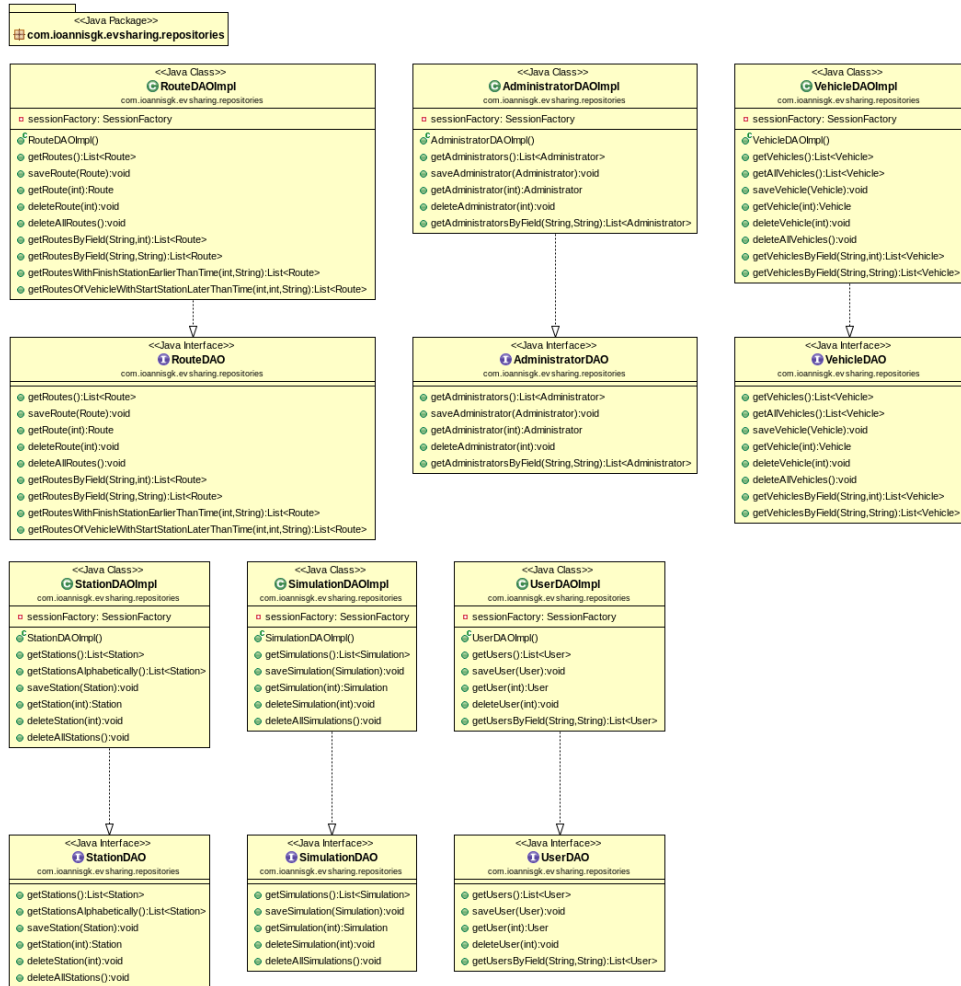


Figure 9.3.3: Class diagram for Repositories package.

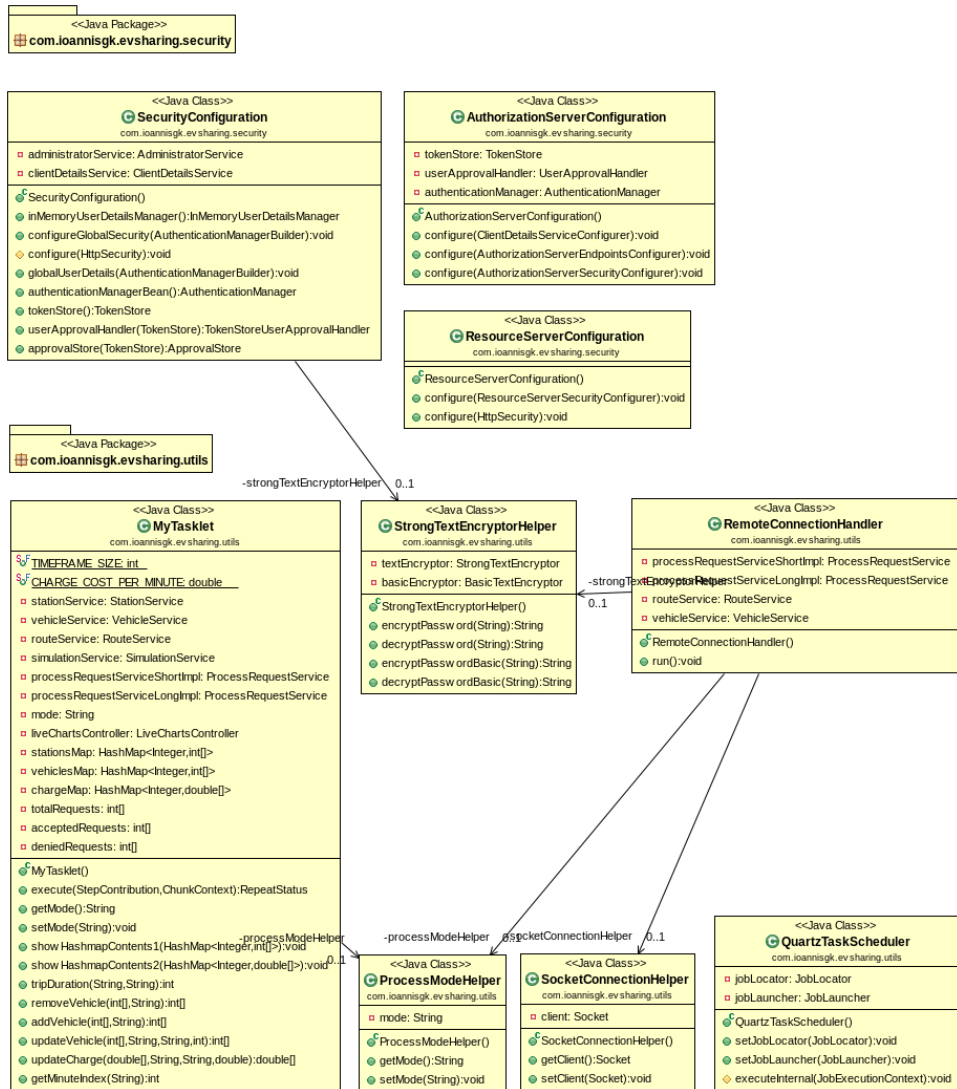


Figure 9.3.4: Class diagram for Security and Utils packages.

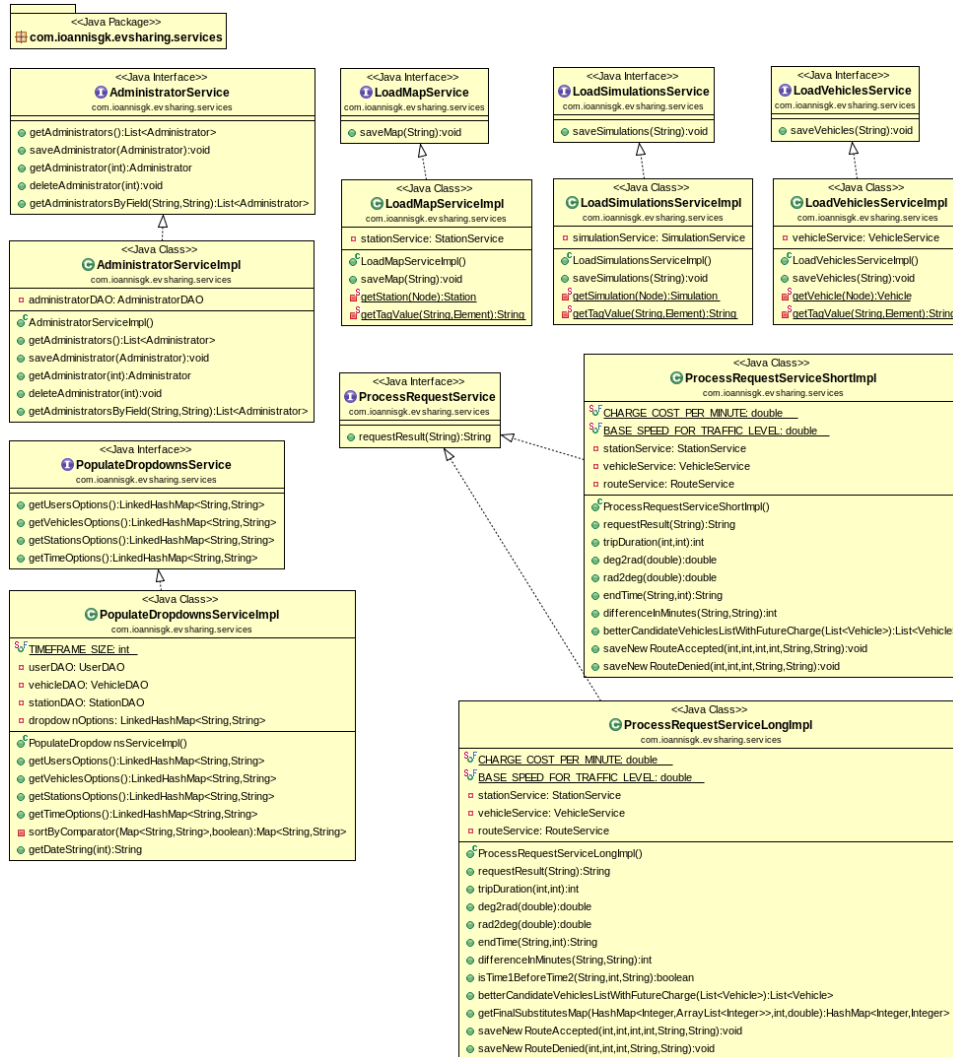


Figure 9.3.5: Class diagram for Services package (part I).

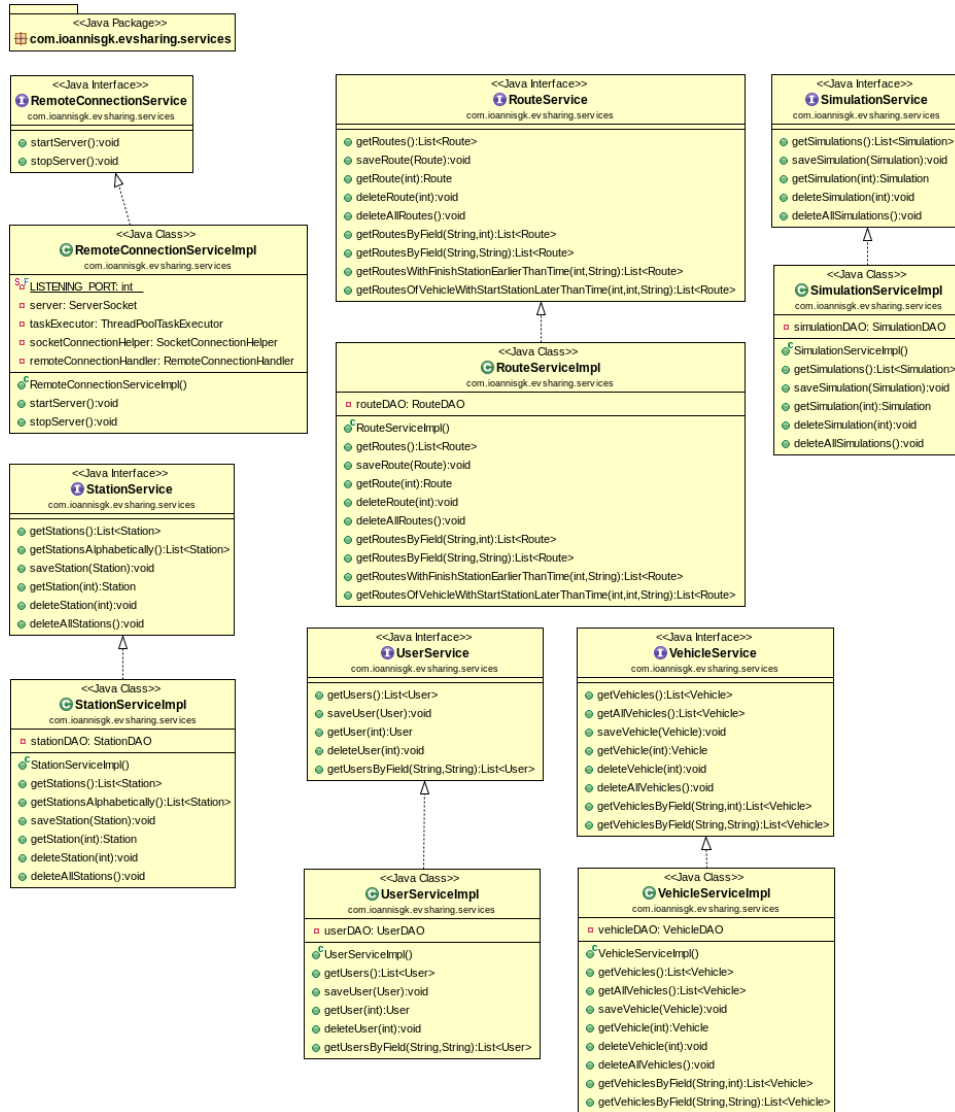


Figure 9.3.6: Class diagram for Services package (part II).

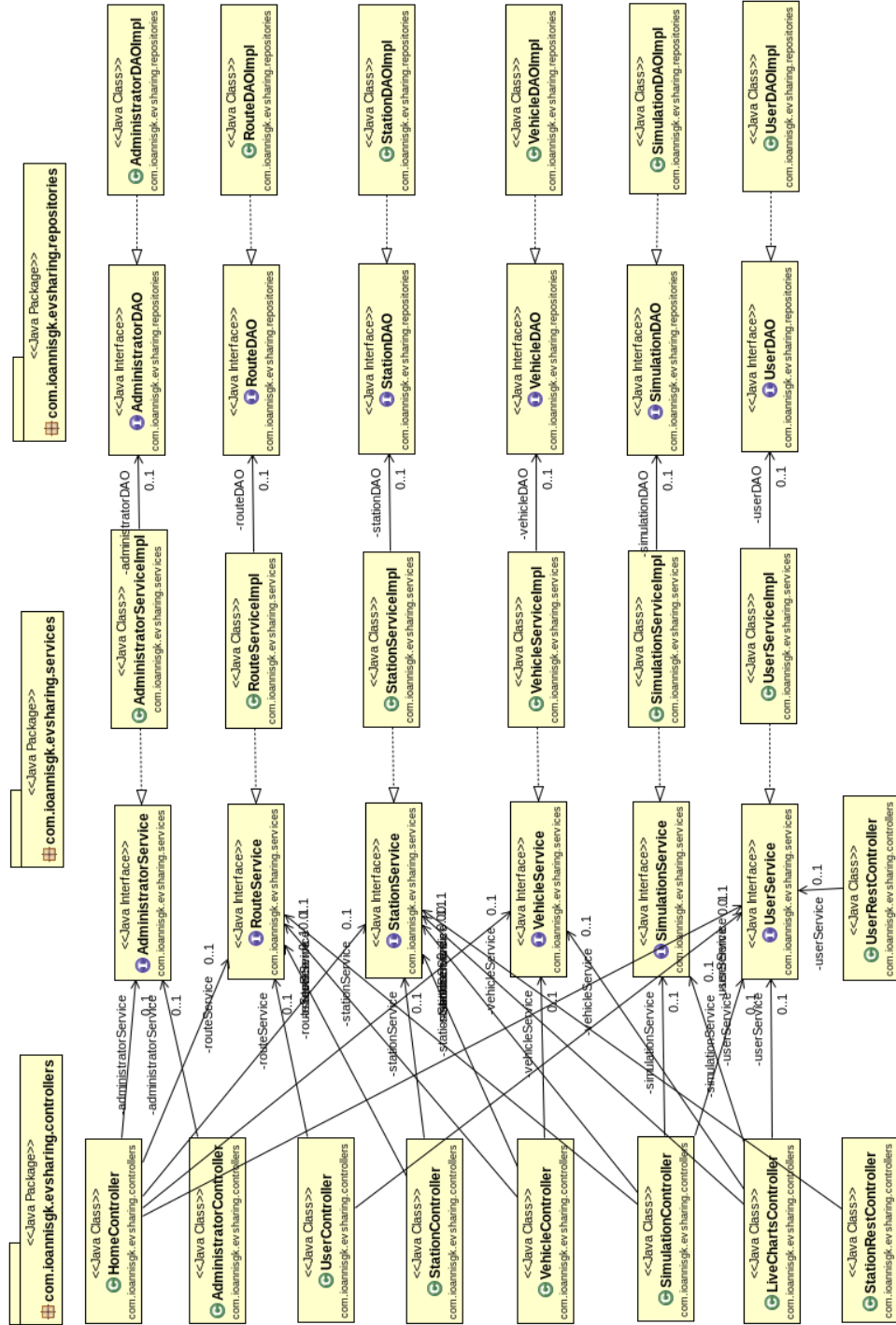


Figure 9.3.7: Class diagram that shows the Controllers, Entities and Repositories classes and their relationships.

The web platform class diagrams depict the classes that are inside each packages: Controllers (Fig. 9.3.1), Entities (Fig. 9.3.2), Repositories (Fig. 9.3.3), Security (Fig. 9.3.4), Utils (Fig. 9.3.4) and Services (Fig. 9.3.5 and Fig. 9.3.6). The last class diagram shows the Controllers, Entities and Repositories classes and their relationships (Fig. 9.3.7).

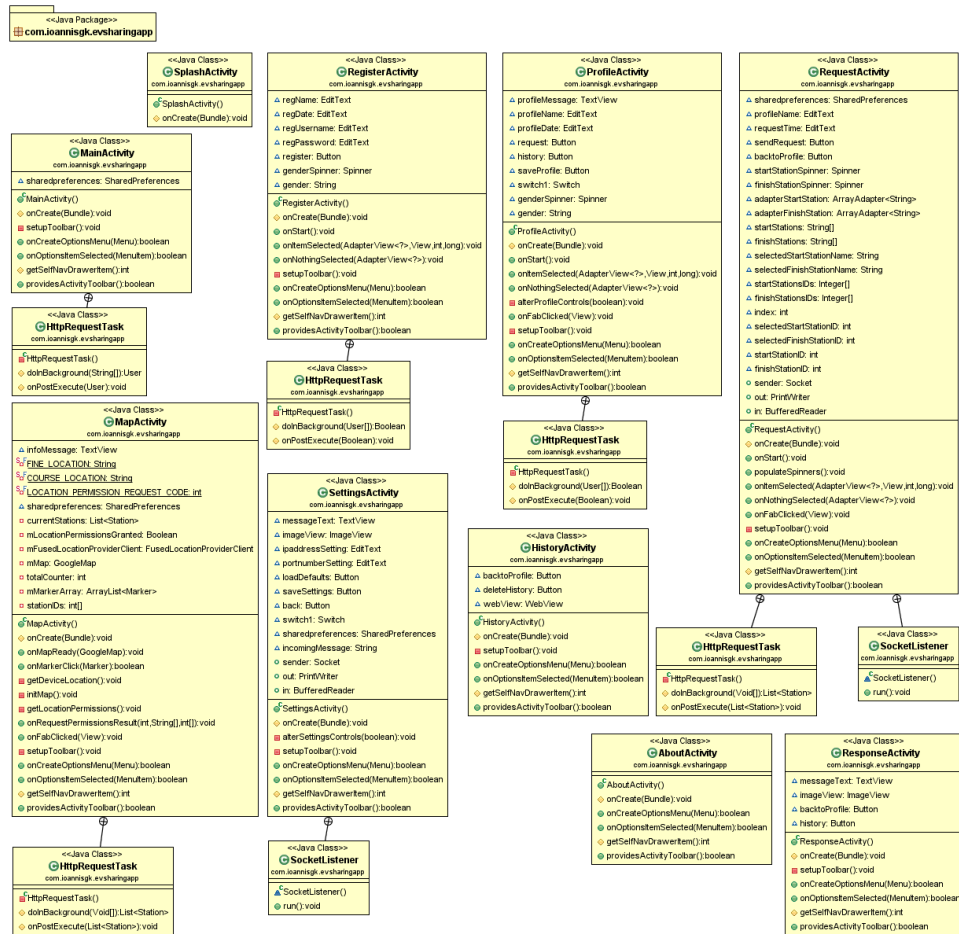


Figure 9.4.1: Class diagram for Evsharingapp package.

9.4 MOBILE APPLICATION CLASS DIAGRAMS

The mobile application class diagrams depict the classes that are inside each packages: Evsharingapp (Fig. 9.4.1), Base (Fig. 9.4.2), Entities (Fig. 9.4.2) and Utils (Fig. 9.4.3).

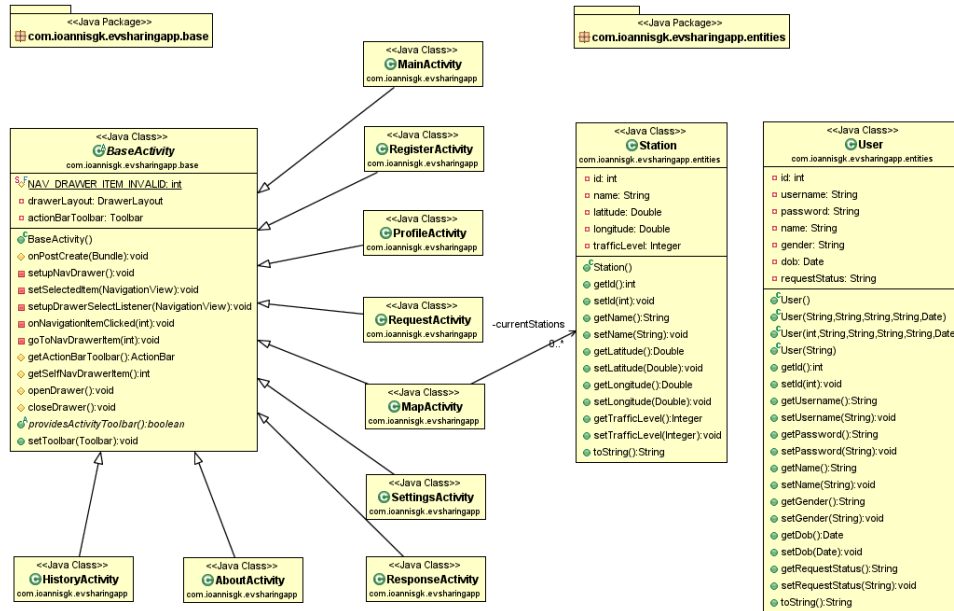


Figure 9.4.2: Class diagram for Base and Entities packages.

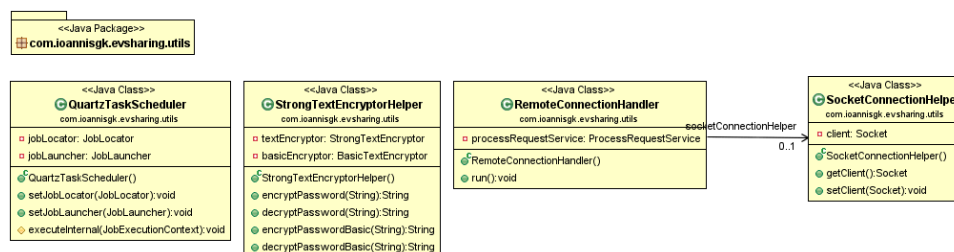


Figure 9.4.3: Class diagram for Utils package.

9.5 SUMMARY OF TEST CASES

A summary of functional and non-functional test cases for the web platform and the mobile application is presented in Tables 9.5.1 and 9.5.2 respectively. All test case results are characterized as "Passed".

PAGE	TEST CASE	RESULT
1. Login Page	Administrator login	PASS
2. Dashboard Panel	Access routes data/delete routes	PASS
3. Add New Route	Add a new route	PASS
4. Update Route	Update existing route data	PASS
5. Manage Admins	Access admins data/delete admins	PASS
6. Add New Admin	Add a new admin	PASS
7. Update Admin	Update existing admin data	PASS
8. Manage Users	Access users data/delete users	PASS
9. Add New User	Add a new user	PASS
10. Update User	Update existing user data	PASS
11. Manage Stations	Access stations data/upload XML	PASS
12. Add New Station	Add a new station	PASS
13. Update Station	Update existing station data	PASS
14. Manage Vehicles	Access vehicles data/upload XML	PASS
15. Add New Vehicle	Add a new vehicle	PASS
16. Update Vehicle	Update existing vehicle data	PASS
17. Simulation Panel	Access simulation data/upload XML	PASS
18. Live Charts	Access live charts and statistics	PASS
(security)	Protect against SQL Injection	PASS
(security)	Store hashed values of passwords	PASS
(performance)	Fast load and response rate	PASS
(performance)	Offer uninterrupted service to users	PASS

Table 9.5.1: Summary of functional and non-functional test cases for the web platform.

SCREEN	TEST CASE	RESULT
1. Main Login	User login	PASS
2. User Registration	Add a new account	PASS
3. My Profile Screen	Access/edit profile details	PASS
4. User History	Access/delete history of requests	PASS
5. Open Map Screen	Get and select start/finish stations	PASS
6. Request Screen	Send request, get and save response	PASS
7. Settings Screen	Change the application settings	PASS
(security)	Send/receive encrypted messages	PASS
(security)	Authenticate users on requests	PASS
(performance)	Fast load and response rate	PASS
(performance)	Offer uninterrupted service to users	PASS

Table 9.5.2: Summary of functional and non-functional test cases for the mobile application.

9.6 ROUTE GENERATOR RESULTS

In order to better evaluate the behavior of the system, we developed a tool to generate random routes based on different criteria. Those criteria are:

- The maximum number of route requests in a day
- The maximum amount of minutes that a user is allowed to request a vehicle in the future
- The start times density factor, that essentially means how "close" or how "further away" in time, the start times of the route requests are

Detailed results are presented in Fig. 9.6.1 and Fig. 9.6.2.

Max time between	Requests allocat	Short mode	Long mode		Max time between	Short mode	Long mode	
60	0.18	31.67	33.33		60	41.334	43	
60	0.36	58.33	58.33		120	43.668	44.666	
60	0.54	35	35		180	41.002	42.334	
60	0.72	40	41.67		240	47.668	48.002	
60	0.9	41.67	46.67		300	50	50.334	
120	0.18	40	43.33		360	46.334	47.668	
120	0.36	51.67	53.33		420	46.668	47.666	
120	0.54	50	50		480	50.666	53.332	
120	0.72	36.67	36.67		540	54.332	55.332	Average gain %:
120	0.9	40	40		600	50.668	51.668	1.1662
180	0.18	36.67	36.67					
180	0.36	46.67	48.33		Requests allocat	Short mode	Long mode	
180	0.54	43.33	48.33		0.18	46.501	47.334	
180	0.72	41.67	41.67		0.36	50.333	51.332	
180	0.9	36.67	36.67		0.54	49.666	51.666	
240	0.18	51.67	51.67		0.72	45.669	46.335	
240	0.36	50	50		0.9	44.001	45.334	
240	0.54	55	55					
240	0.72	36.67	36.67					
240	0.9	45	46.67					
300	0.18	58.33	58.33					
300	0.36	50	50					
300	0.54	58.33	60					
300	0.72	41.67	41.67					
300	0.9	41.67	41.67					
360	0.18	51.67	51.67					
360	0.36	45	45					
360	0.54	48.33	51.67					
360	0.72	46.67	46.67					
360	0.9	40	43.33					
420	0.18	51.67	51.67					
420	0.36	48.33	48.33					
420	0.54	36.67	40					
420	0.72	56.67	58.33					
420	0.9	40	40					
480	0.18	43.33	46.67					
480	0.36	50	53.33					
480	0.54	58.33	63.33					
480	0.72	55	55					
480	0.9	46.67	48.33					
540	0.18	45	45					
540	0.36	58.33	60					
540	0.54	55	55					
540	0.72	50	53.33					
540	0.9	63.33	63.33					
600	0.18	55	55					
600	0.36	45	46.67					
600	0.54	56.67	58.33					
600	0.72	51.67	51.67					
600	0.9	45	46.67					

Figure 9.6.1: Detailed efficiency rate results for 60 requests per day.

Max time between	Requests allocat	Short mode	Long mode		Max time between	Short mode	Long mode	
60	0.18	30	30.83		60	27.666	27.832	
60	0.36	26.67	26.67		120	34.666	37.5	
60	0.54	25	25		180	42.334	43.166	
60	0.72	33.33	33.33		240	43	45.834	
60	0.9	23.33	23.33		300	40.166	41.832	
120	0.18	28.33	28.33		360	43.498	45.166	
120	0.36	40.83	45		420	45.5	47.832	
120	0.54	42.5	45		480	43.166	45.832	
120	0.72	32.5	40		540	42.5	43.666	Average gain %:
120	0.9	29.17	29.17		600	43.166	45.332	1.833
180	0.18	49.17	52.5					
180	0.36	46.67	46.67		Requests allocat	Short mode	Long mode	
180	0.54	45.83	45.83		0.18	44.833	46.416	
180	0.72	41.67	42.5		0.36	42.667	44.832	
180	0.9	28.33	28.33		0.54	42.999	44.166	
240	0.18	48.33	49.17		0.72	37.666	40.333	
240	0.36	50	55.83		0.9	34.666	36.249	
240	0.54	44.17	44.17					
240	0.72	41.67	45					
240	0.9	30.83	35					
300	0.18	45	45					
300	0.36	41.67	45.83					
300	0.54	45.83	48.33					
300	0.72	33.33	34.17					
300	0.9	35	35.83					
360	0.18	53.33	54.17					
360	0.36	43.33	47.5					
360	0.54	43.33	43.33					
360	0.72	37.5	37.5					
360	0.9	40	43.33					
420	0.18	45.83	45.83					
420	0.36	35	35.83					
420	0.54	56.67	58.33					
420	0.72	45.83	52.5					
420	0.9	44.17	46.67					
480	0.18	41.67	47.5					
480	0.36	50.83	50.83					
480	0.54	47.5	48.33					
480	0.72	32.5	36.67					
480	0.9	43.33	45.83					
540	0.18	56.67	57.5					
540	0.36	45	45.83					
540	0.54	38.33	39.17					
540	0.72	37.5	38.33					
540	0.9	35	37.5					
600	0.18	50	53.33					
600	0.36	46.67	48.33					
600	0.54	40.83	44.17					
600	0.72	40.83	43.33					
600	0.9	37.5	37.5					

Figure 9.6.2: Detailed efficiency rate results for 120 requests per day.

9.7 VIRTUAL DEMO

9.8 PRESENTATION

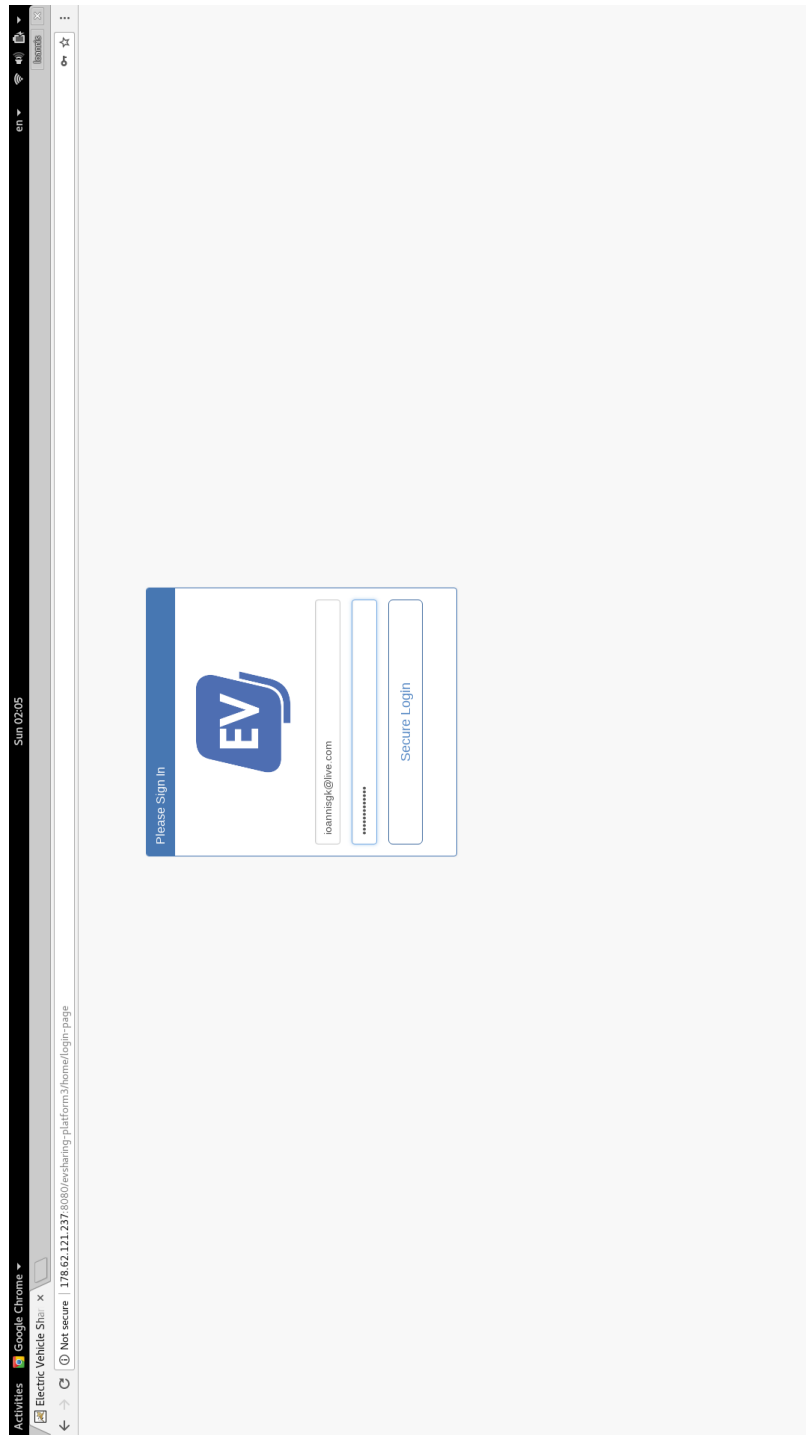


Figure 9.7.1: Administrator logging in to the web platform.

EV Sharing Platform v1.0

Dashboard

Live Charts

Admins

Users

Stations

Vehicles

Simulation

Logout

EV

Dashboard Panel

Map satellite

63 Total Routes!

View Details

9 Total Users!

View Details

18 Total Vehicles!

View Details

27 Total Stations!

View Details

Info Panel

The map shows the stations from the database, and routes can be created or updated.

Add Routes

Add New Route

System Status

Short Term Mode

START SERVICE

SERVICE IS STOPPED

STOP SERVICE

Notifications Panel

Requests Accepted: 63

Requests Denied: 57

Efficiency Rate: 52.5%

Accepted Routes Details

Start Station	Finish Station	Start Time	End Time	Username	License Plates	Actions
Brislington Park and Ride 1	Avon Fire and Rescue	06:09:00	06:31:00	user44@test.com	BRTED010	Update Delete
Avon Fire and Rescue	Childrens Scrapstore	06:37:00	06:48:00	user66@test.com	BRM00005	Update Delete
Temple Gate Car Park	Temple Gate Car Park	06:44:00	06:44:00	user11@test.com	BRM00015	Update Delete
College Street Car Park	College Street Car Park	06:49:00	06:49:00	user11@test.com	BRHY0008	Update Delete
Millenium Square Car Park	Millenium Square Car Park	06:53:00	06:53:00	user33@test.com	BRFF0016	Update Delete
The Grove Car Park	Avon Fire and Rescue	07:01:00	07:06:00	user77@test.com	BRFF0006	Update Delete
Childrens Scrapstore	Southmead Hospital	07:06:00	07:16:00	user22@test.com	BRN00003	Update Delete
Avon Fire and Rescue	The Grove Car Park	07:08:00	07:13:00	user33@test.com	BRTED010	Update Delete

Figure 9.7.2: Dashboard Panel page of the web platform.

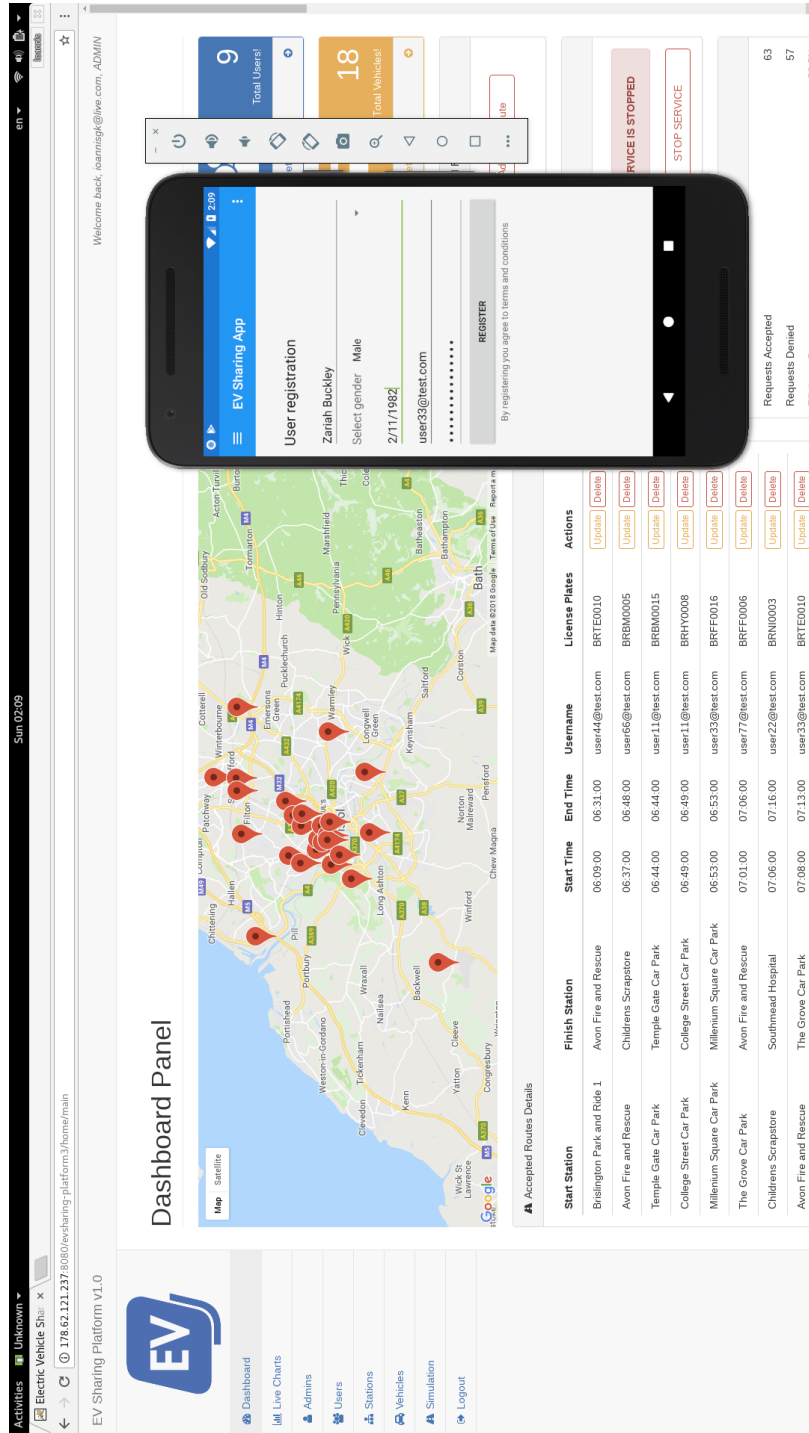


Figure 9.7.3: User registration screen on the mobile application.

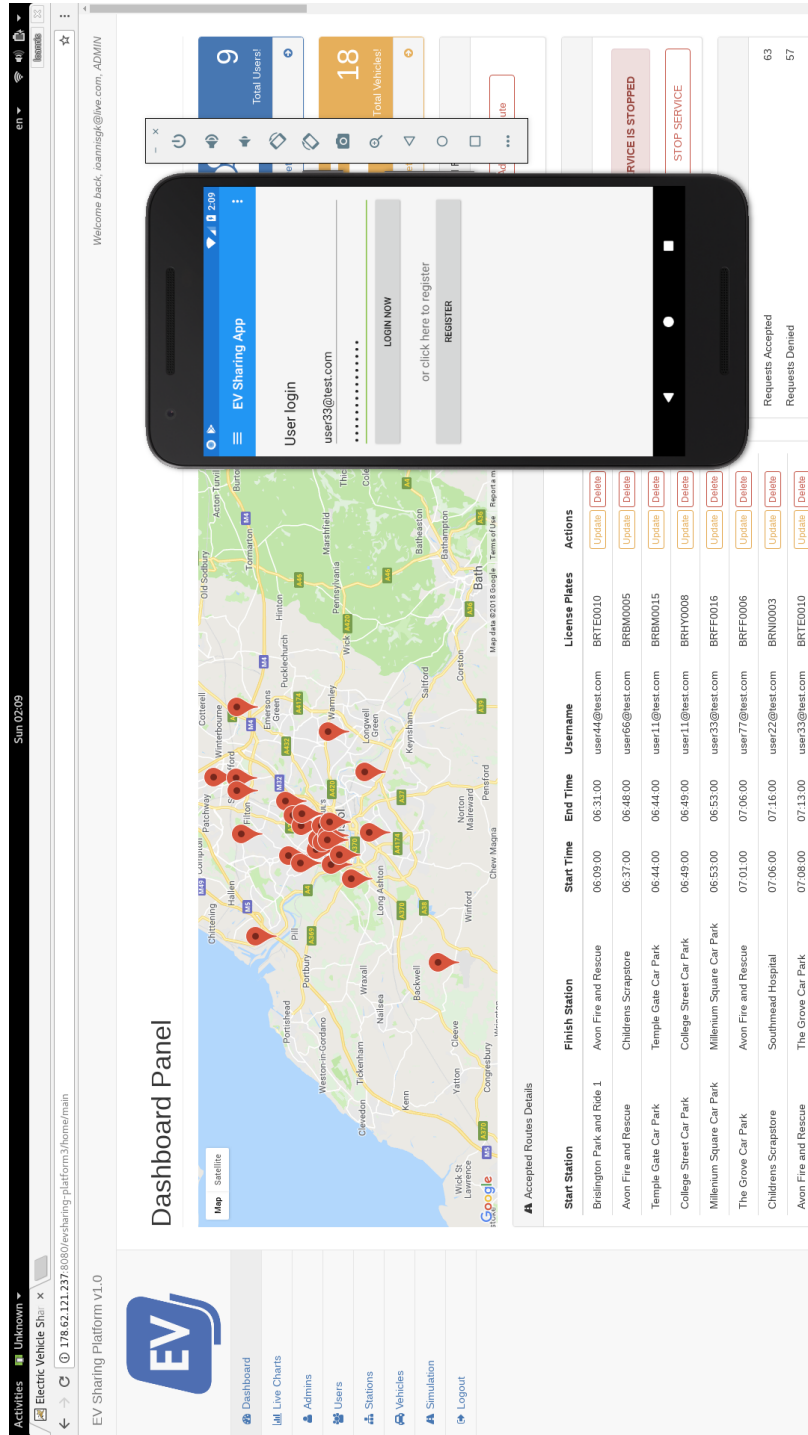


Figure 9.7.4: User login screen on the mobile application.

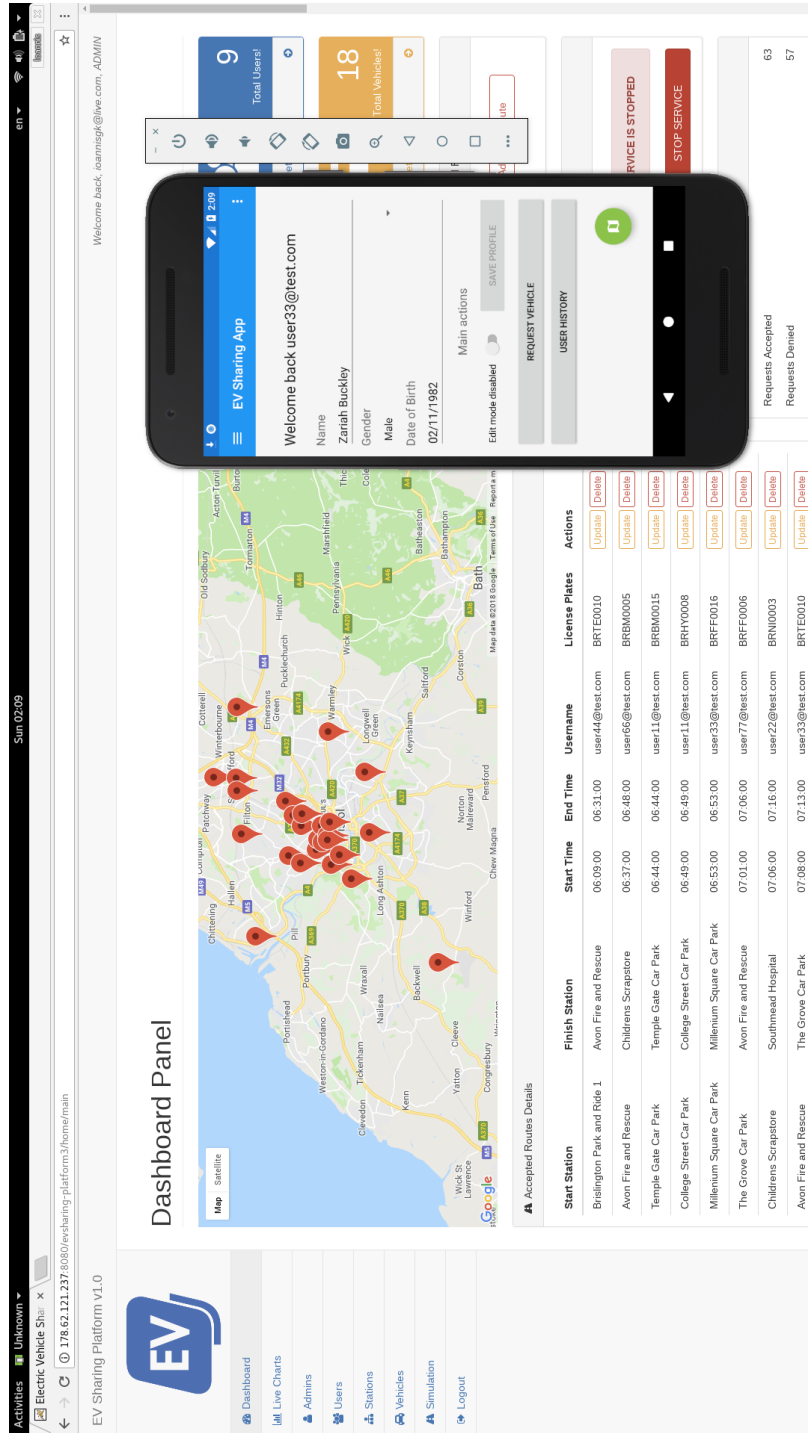


Figure 9.7.5: Profile screen on the mobile application.

Welcome back, iourmsg@live.com, ADMIN

en

Sun 02:24

Electric Vehicle Share

EV Sharing Platform v1.0

Dashboard

Live Charts

Admins

Users

Stations

Vehicles

Simulation

Logout

Dashboard Panel

Map satellite

Accepted Routes Details

Start Station	Finish Station	Username	Start Time	End Time	License Plates	Actions
Brislington Park and Ride 1	Avon Fire and Rescue	user44@rest.com	06:09:00	06:31:00	BRE010	Update Delete
Avon Fire and Rescue	Childrens Scrapstore	user66@rest.com	06:37:00	06:48:00	BRM0005	Update Delete
Temple Gate Car Park	Temple Gate Car Park	user11@rest.com	06:44:00	06:44:00	BRM0015	Update Delete
College Street Car Park	College Street Car Park	user11@rest.com	06:49:00	06:49:00	BRM0008	Update Delete
Millenium Square Car Park	Millenium Square Car Park	user33@rest.com	06:53:00	06:53:00	BRFF0016	Update Delete
The Grove Car Park	Avon Fire and Rescue	user77@rest.com	07:01:00	07:06:00	BRFF0006	Update Delete
Childrens Scrapstore	Southmead Hospital	user22@rest.com	07:06:00	07:16:00	BRN0003	Update Delete
Avon Fire and Rescue	The Grove Car Park	user33@rest.com	07:08:00	07:13:00	BRE010	Update Delete

63
Requests Accepted
57
Requests Denied
52.5%

EV Sharing App

Destination: IKEA Bristol

9 Total Users!

18 Total Vehicles!

STOP SERVICE

Figure 9.7.6: Open Map screen on the mobile application.

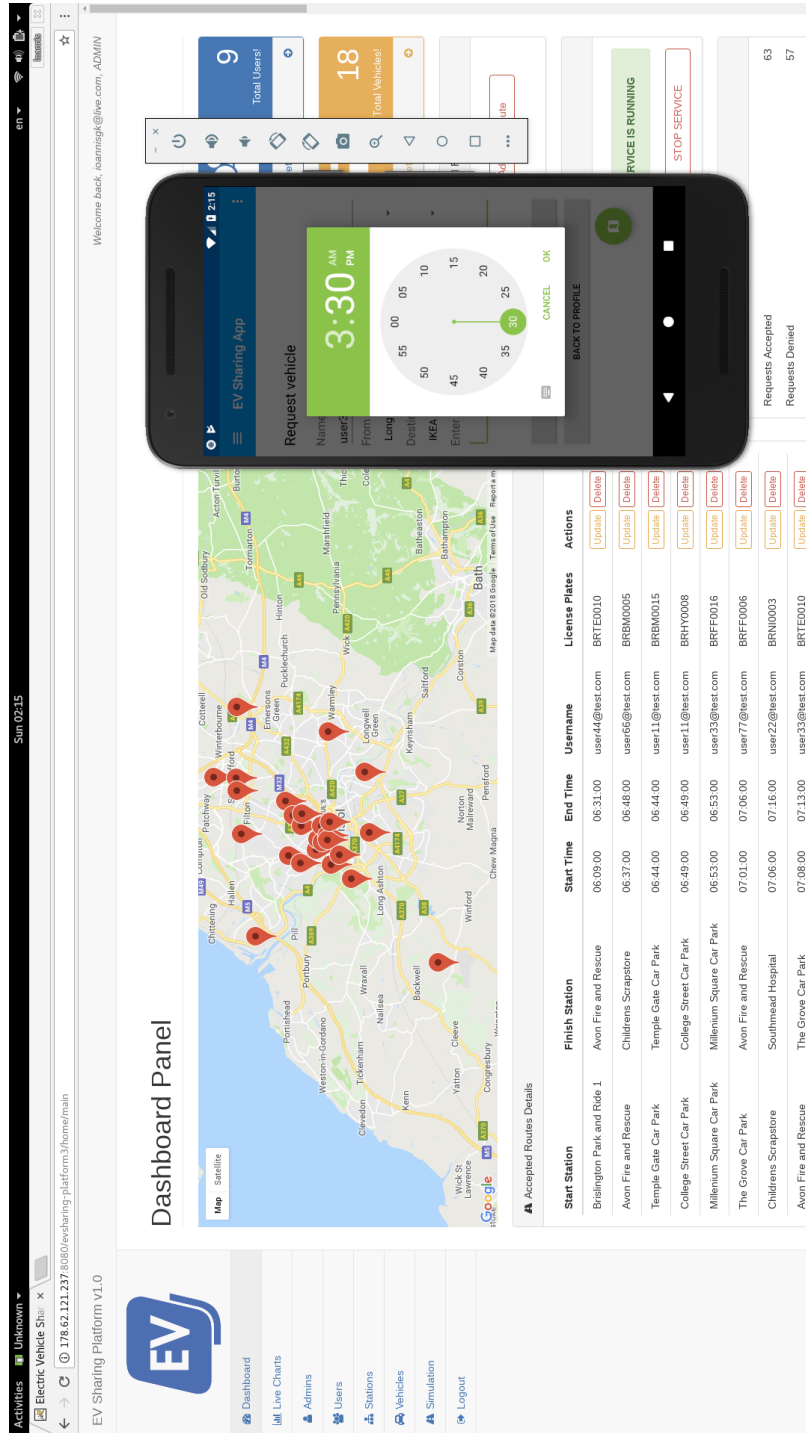


Figure 9.7.7: Selecting request time on the mobile application.

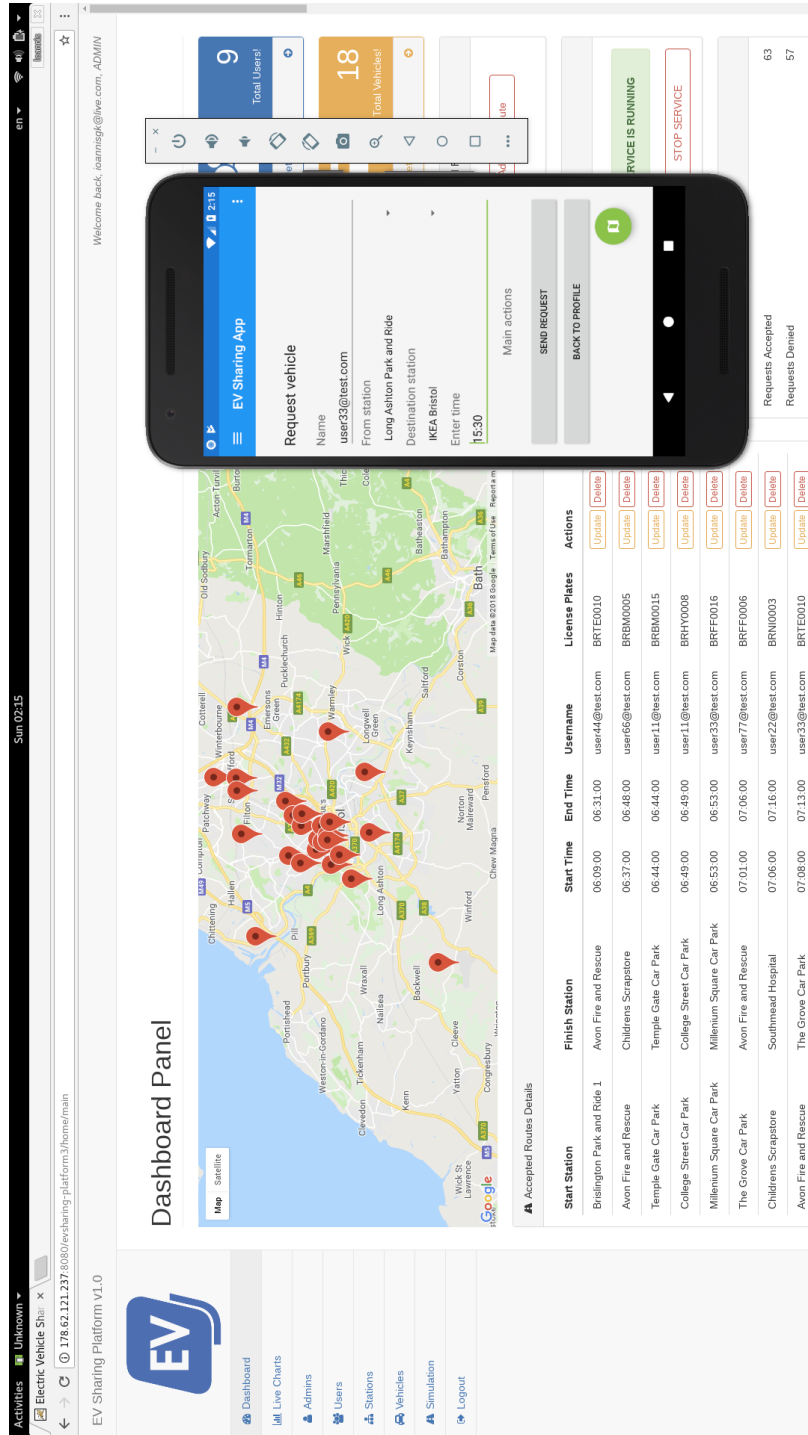


Figure 9.7.8: Request vehicle screen on the mobile application.

EV Sharing Platform v1.0

Dashboard

Live Charts

Admins

Users

Stations

Vehicles

Simulation

Logout

EV

Accepted Routes Details

Start Station	Finish Station	Start Time	End Time	Username	License Plates	Actions
Brislington Park and Ride 1	Avon Fire and Rescue	06:09:00	06:31:00	user44@test.com	BRTED010	Update Delete
Avon Fire and Rescue	Childrens Scrapstore	06:37:00	06:48:00	user66@test.com	BRM00005	Update Delete
Temple Gate Car Park	Temple Gate Car Park	06:44:00	06:44:00	user11@test.com	BRM00015	Update Delete
College Street Car Park	College Street Car Park	06:49:00	06:49:00	user11@test.com	BRHY0008	Update Delete
Millenium Square Car Park	Millenium Square Car Park	06:53:00	06:53:00	user33@test.com	BRFF0016	Update Delete
The Grove Car Park	Avon Fire and Rescue	07:01:00	07:06:00	user77@test.com	BRFF0006	Update Delete
Childrens Scrapstore	Southmead Hospital	07:06:00	07:16:00	user22@test.com	BRN00003	Update Delete
Avon Fire and Rescue	The Grove Car Park	07:08:00	07:13:00	user33@test.com	BRTED010	Update Delete

Requests Accepted: 63
Requests Denied: 57
Efficiency Rate: 52.5%

Dashboard Panel

EV Sharing App

Vehicle request

Your request is accepted and confirmed

Main actions

BACK TO PROFILE

USER HISTORY

VEHICLE IS RUNNING

STOP SERVICE

9 Total Users!

18 Total Vehicles!

400

en

Sun 02:15

Electric Vehicle Share

178.62.121.237:8080/evsharing-platform-3/home/main

Welcome back, iourmsgl@live.com, ADMIN

Figure 9.7.9: Receiving request status on the mobile application.

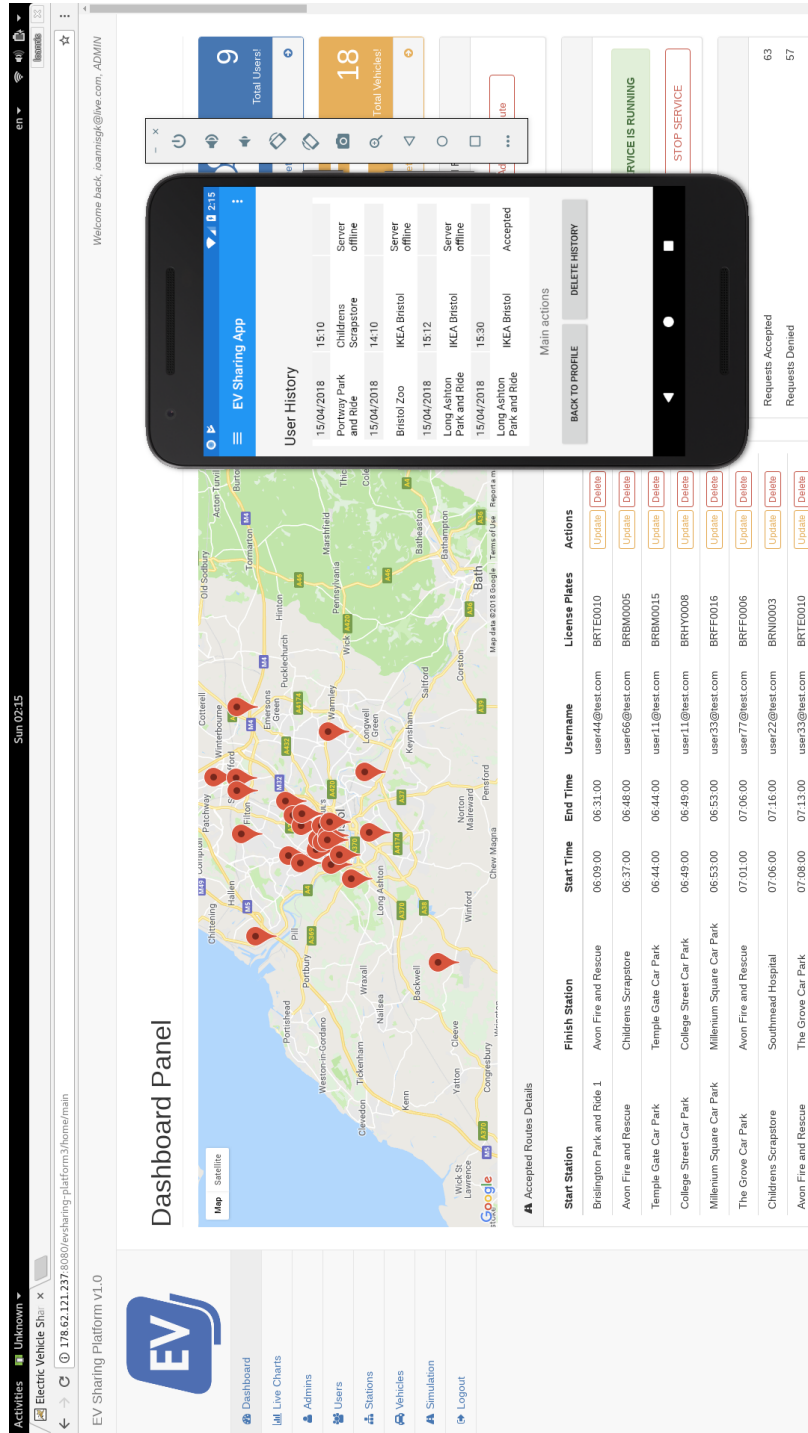


Figure 9.7.10: User History screen on the mobile application.

Activities Google Chrome en Sun 02:36

Electric Vehicle Share x

178.62.121.237:8080/ev-sharing-platform:3/home/main

Avon Fire and Rescue	Create Centre	08:29:00	08:43:00	user99@test.com	BRBM0005	Update	Delete
Wessex Garages Bristol	Penway Park and Ride	08:32:00	08:56:00	user66@test.com	BRK00014	Update	Delete
The Grove Car Park	Knowle West Media Centre	08:33:00	08:46:00	user77@test.com	BRT00010	Update	Delete
Bristol Zoo	Trenchard Street Car Park	08:36:00	08:44:00	user33@test.com	BRVW00017	Update	Delete
The Grove Car Park	Trenchard Street Car Park	08:39:00	08:43:00	user11@test.com	BRBM00005	Update	Delete
Nissan Wessex Bristol	Bristol Zoo	08:39:00	08:47:00	user11@test.com	BRN00013	Update	Delete
Simplyhealth House	Childrens Scrapstore	08:40:00	08:49:00	user77@test.com	BRFF0006	Update	Delete
Millenium Square Car Park	West End Car Park	08:41:00	08:46:00	user33@test.com	BRFF0016	Update	Delete
The Grove Car Park	Millenium Square Car Park	08:42:00	08:44:00	user88@test.com	BRBM0015	Update	Delete
Avon Fire and Rescue	Childrens Scrapstore	08:43:00	08:54:00	user33@test.com	BRSM0012	Update	Delete
Avon Fire and Rescue	Create Centre	08:52:00	09:06:00	user66@test.com	BRBM00005	Update	Delete
Childrens Scrapstore	Bristol Airport	08:53:00	09:30:00	user22@test.com	BRBM0015	Update	Delete
The Grove Car Park	Childrens Scrapstore	08:53:00	09:07:00	user77@test.com	BRFF0006	Update	Delete
Create Centre	Southmead Hospital	08:53:00	09:10:00	user22@test.com	BRSM0002	Update	Delete
Simplyhealth House	West End Car Park	08:54:00	09:02:00	user88@test.com	BRSM0012	Update	Delete
College Street Car Park	West End Car Park	08:54:00	08:57:00	user33@test.com	BRHY0008	Update	Delete
Brislington Park and Ride 1	Avon Fire and Rescue	08:54:00	09:16:00	user11@test.com	BRT00010	Update	Delete
West End Car Park	Tobacco Factory	08:55:00	09:02:00	user11@test.com	BRHY0018	Update	Delete
Knowle West Media Centre	Trenchard Street Car Park	08:55:00	09:13:00	user22@test.com	BRVW0007	Update	Delete
Trenchard Street Car Park	Brislington Park and Ride 2	08:59:00	09:17:00	user88@test.com	BRK00004	Update	Delete
West End Car Park	Create Centre	09:00:00	09:07:00	user11@test.com	BRHY0008	Update	Delete
Long Ashton Park and Ride	IKEA Bristol	15:30:00	15:48:00	user33@test.com	BRVW0007	Update	Delete

Copyright © Ioannis Skourbounis, 2018

Figure 9.7.11: The request saved as a route in the Dashboard Panel of the web platform.

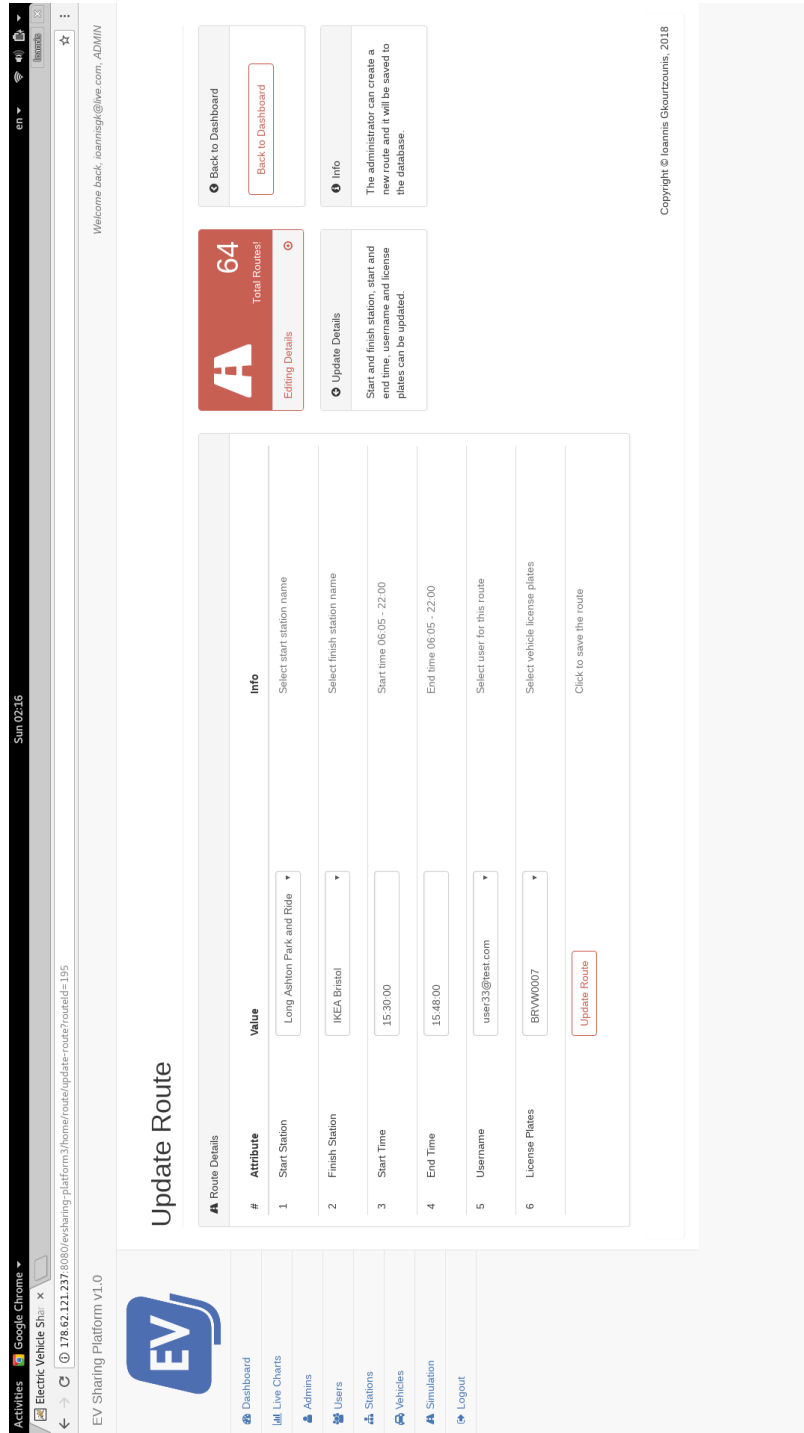


Figure 9.7.12: Updating route details in the web platform.

EV Sharing Platform v1.0

Admin Accounts Details

Full Name	Username	Password	Role	Actions
Ioannis Gkoutzounis	ioannsgf@live.com	*****	ADMIN	Update Delete
Janice Chamberlain	admin1@east.com	*****	ADMIN	Update Delete
Thomas Cunningham	admin2@east.com	*****	ADMIN	Update Delete
David Charles	admin3@east.com	*****	MODERATOR	Update Delete
Juan Segundo	admin4@east.com	*****	MODERATOR	Update Delete

Manage Admins

5 Total Admins
Viewing Details

Add Admins
Add New Admin

Info
This panel is protected and only visible to logged in admins, not moderators.

Manage Admins
Admins names, usernames and roles are shown and they can be created or updated.

Copyright © Ioannis Gkoutzounis, 2018

Figure 9.7.13: Managing Admin accounts in the web platform.

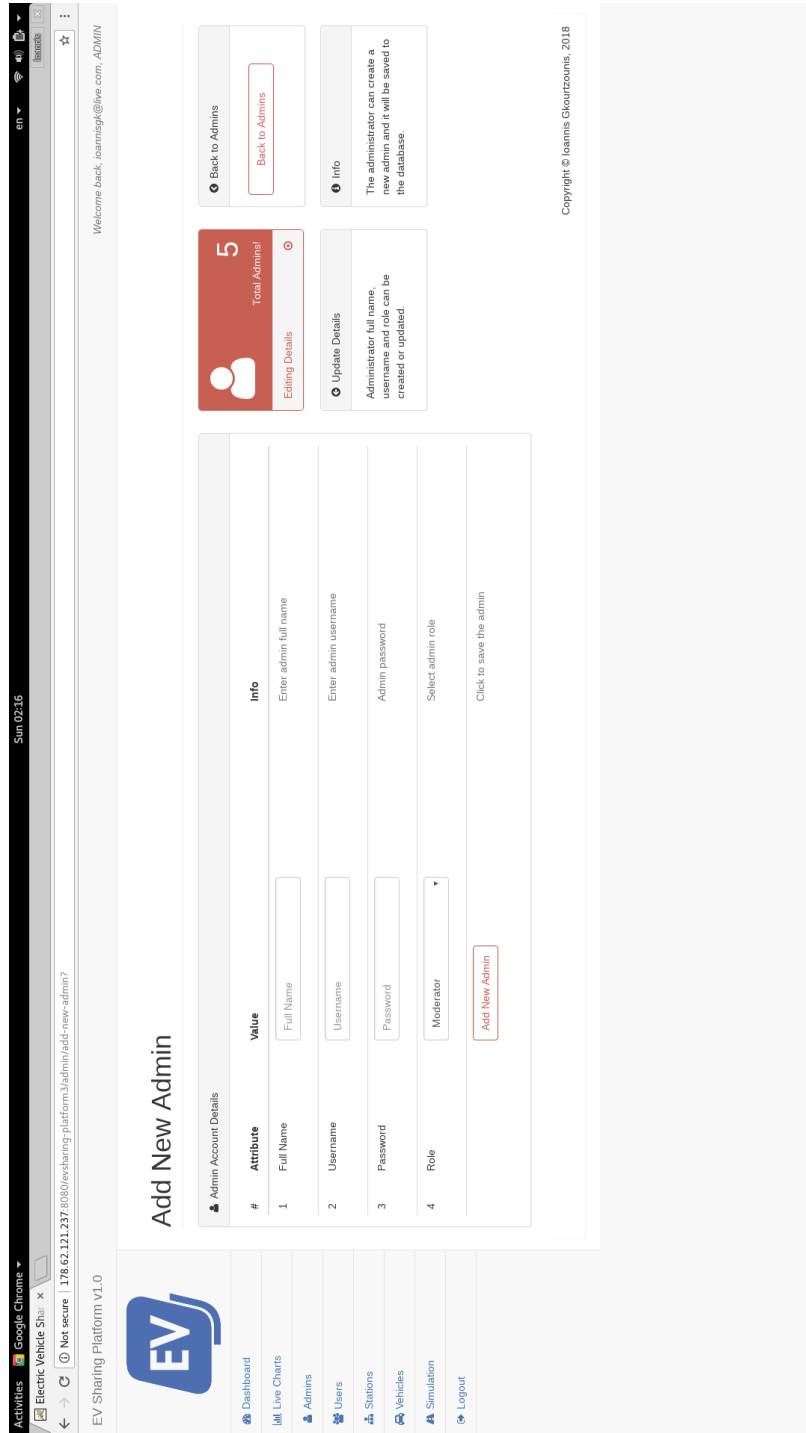


Figure 9.7.14: Adding a new Admin account in the web platform.

Activities Google Chrome Electric Vehicle Share x Sun 02:16

178.62.121.237:8080/ev-sharing-platform-3/user/list

Welcome back, loannisgr@hve.com, ADMIN

EV

- Dashboard
- Live Charts
- Admins
- Users**
- Stations
- Vehicles
- Simulation
- Logout

Manage Users

9 Total Users | Viewing Details

Add Users

Info
This panel is protected and visible to all logged in admins and moderators.

Manage Users
Users usernames, gender and dates of birth are shown and they can be created or updated.

Full Name	Username	Gender	Date of Birth	Actions
Pierce Bell	user11@hve.com	Male	02/11/1990	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Breanna Solis	user22@hve.com	Female	02/11/1981	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Zariah Buckley	user33@hve.com	Male	02/11/1982	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Julius Carson	user44@hve.com	Female	02/11/1983	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Kylan Tyler	user55@hve.com	Male	02/11/1984	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Zaire Craig	user66@hve.com	Female	02/11/1985	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Melvin Nielson	user77@hve.com	Male	02/11/1986	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Mattie Bennett	user88@hve.com	Female	02/11/1987	<input type="button" value="Update"/> <input type="button" value="Delete"/>
Brett Marshall	user99@hve.com	Male	02/11/1988	<input type="button" value="Update"/> <input type="button" value="Delete"/>

Copyright © Ioannis Ghourzounis, 2018

Figure 9.7.15: Managing user accounts in the web platform.

en Sun 02:16

178.62.121.237:8080/ev-sharing-platform-3/station/list

Welcome back, iourmsgk@live.com, ADMIN

Manage Stations

27 Total Stations!

Viewing Details

Manage Stations
Station names, coordinates and traffic levels are shown and they can be created or updated. This panel is protected.

Add Stations
Add New Station

Upload XML
Choose File No file chosen
Upload XML

Station Name	Latitude	Longitude	Traffic Level	Actions
Avon Fire and Rescue	51.452937	-2.59727	5	Update Delete
Brislington Park and Ride 1	51.428051	-2.537584	2	Update Delete
Brislington Park and Ride 2	51.428141	-2.537584	1	Update Delete
Bristol Airport	51.38686	-2.712836	1	Update Delete
Bristol and Bath Science Park	51.500883	-2.478359	1	Update Delete
Bristol Parkway Station	51.514366	-2.542644	1	Update Delete
Bristol Zoo	51.464625	-2.621863	2	Update Delete
Childrens Scrapstore	51.46919	-2.578083	3	Update Delete
College Street Car Park	51.452485	-2.602735	4	Update Delete
Create Centre	51.447315	-2.62305	3	Update Delete
Hewlett Packard Bristol Office	51.501078	-2.554901	1	Update Delete
IKEA Bristol	51.473057	-2.564405	2	Update Delete
Knowle West Media Centre	51.426131	-2.59339	2	Update Delete
Long Ashton Park and Ride	51.43881	-2.636174	2	Update Delete
Longwell Green Leisure Centre	51.449065	-2.500966	1	Update Delete
Millenium Square Car Park	51.449286	-2.600439	4	Update Delete
Mortality Bristol	51.501557	-2.543303	1	Update Delete
Nissan Wessex Bristol	51.464275	-2.587624	3	Update Delete

EV Sharing Platform v1.0

- Dashboard
- Live Charts
- Admins
- Users
- Stations
- Vehicles
- Simulation
- Logout

Figure 9.7.16: Managing stations details in the web platform.

Activities Google Chrome Electric Vehicle Share x Sun 02:17

EV Sharing Platform v1.0

Welcome back, iourmsgr@live.com, ADMIN

Manage Vehicles

Vehicles Details

18 Total Vehicles! Viewing Details

Add Vehicles
Add New Vehicle

Upload XML
Choose File No file chosen
Upload XML

Manage Vehicles
Vehicles plates, models and charge levels are shown and they can be created or updated. This panel is protected.

License Plates	Model Name	At Station	Charge %	Actions
BRBM0005	BMW i3	Avon Fire and Rescue	100.0	Update Delete
BRBM0015	BMW i3	Temple Gate Car Park	100.0	Update Delete
BRFF0006	Ford Focus Electric	The Grove Car Park	100.0	Update Delete
BRFF0016	Ford Focus Electric	Millenium Square Car Park	100.0	Update Delete
BRHY0008	Hyundai Ioniq Electric	College Street Car Park	100.0	Update Delete
BRHY0018	Hyundai Ioniq Electric	West End Car Park	100.0	Update Delete
BRKI0004	KIA Soul EV	Trenchard Street Car Park	100.0	Update Delete
BRKI0014	KIA Soul EV	Wessex Garages Bristol	100.0	Update Delete
BRNI0003	Nissan Leaf	Childrens Scrapstore	100.0	Update Delete
BRNI0013	Nissan Leaf	Nissan Wessex Bristol	100.0	Update Delete
BRRE0009	Renault Zoe	Tobacco Factory	100.0	Update Delete
BRSN0002	Smart Fortwo ED	Create Centre	100.0	Update Delete
BRSN0012	Smart Fortwo ED	Simplyhealth House	100.0	Update Delete
BRTD0010	Tesla Model 3	Bristolington Park and Ride 1	100.0	Update Delete
BRVW0001	Volkswagen e-Up!	Knowle West Media Centre	100.0	Update Delete
BRVW0007	Volkswagen e-Golf	Long Ashton Park and Ride	100.0	Update Delete
BRVW0011	Volkswagen e-Up!	IKEA Bristol	100.0	Update Delete
BRVW0017	Volkswagen e-Golf	Bristol Zoo	100.0	Update Delete

Dashboard
Live Charts
Admins
Users
Stations
Vehicles
Simulation
Logout

Figure 9.7.17: Managing vehicles details in the web platform.

EV Sharing Platform v1.0

Simulation Panel

Map satellite

Requests for Routes Details

Request Message	Start Station	Finish Station	Username	Start Time	Action
"17 26 26 06:44"	Temple Gate Car Park	Temple Gate Car Park	user11@test.com	06:44	Delete
"18 5 20 07:36"	IKEA Bristol	Knowle West Media Centre	user22@test.com	07:36	Delete
"19 21 6 08:36"	Bristol Zoo	Trenchard Street Car Park	user33@test.com	08:36	Delete
"20 3 13 06:35"	Meon Valley Bristol	Create Centre	user44@test.com	06:35	Delete
"21 18 10 06:06"	Hewlett Packard Bristol Office	Long Ashton Park and Ride	user55@test.com	06:06	Delete
"22 26 6 06:39"	Temple Gate Car Park	Trenchard Street Car Park	user66@test.com	06:39	Delete
"23 9 23 07:01"	The Grove Car Park	Avon Fire and Rescue	user77@test.com	07:01	Delete
"24 13 6 07:59"	Create Centre	Trenchard Street Car Park	user88@test.com	07:59	Delete

System Status

Short Term Mode
Short Term Mode
Long Term Mode

START SIMULATION

STOP SIMULATION

STOP SIMULATION IS STOPPED

STOP SIMULATION

Notifications Panel

Requests Accepted: 64
Requests Denied: 57
Efficiency Rate: 52.89%

Users

Avon Fire and Rescue
Station ID: 23

Info Panel

The list shows each request details, and requests can be loaded from a file or deleted.

Upload XML

Choose File No file chosen
Upload XML

Stations

Avon Fire and Rescue
Station ID: 23

EV Sharing Platform v1.0

Dashboard
Live Charts
Admins
Users
Stations
Vehicles
Simulation
Logout

EV

Google Chrome
Electric Vehicle Share
Sun 02:19

Welcome back, iourmsgl@live.com, ADMIN

Figure 9.7.18: Simulation Panel page of the web platform.



Figure 9.7.19: Live Charts page of the web platform.



Figure 9.7.20: Selecting a specific station in Live Charts page.



Figure 9.7.21: Selecting a specific timeframe in Live Charts page.



Copyright © Ioannis Skourtsounis, 2018

Figure 9.7.22: The assigned vehicle to the user request is shown as "Traveling" in Live Charts page.

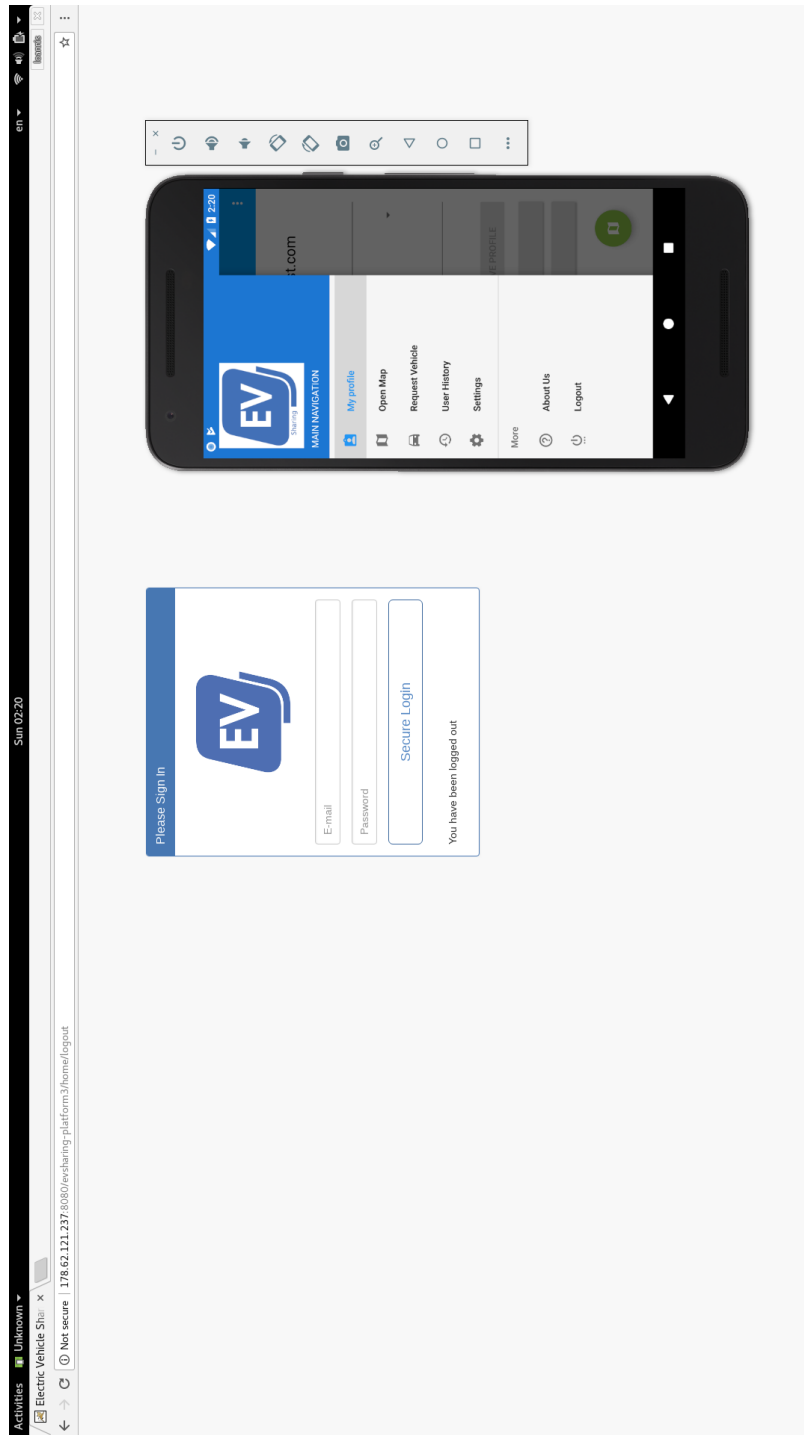


Figure 9.7.23: Navigation menu on the mobile application (part I).

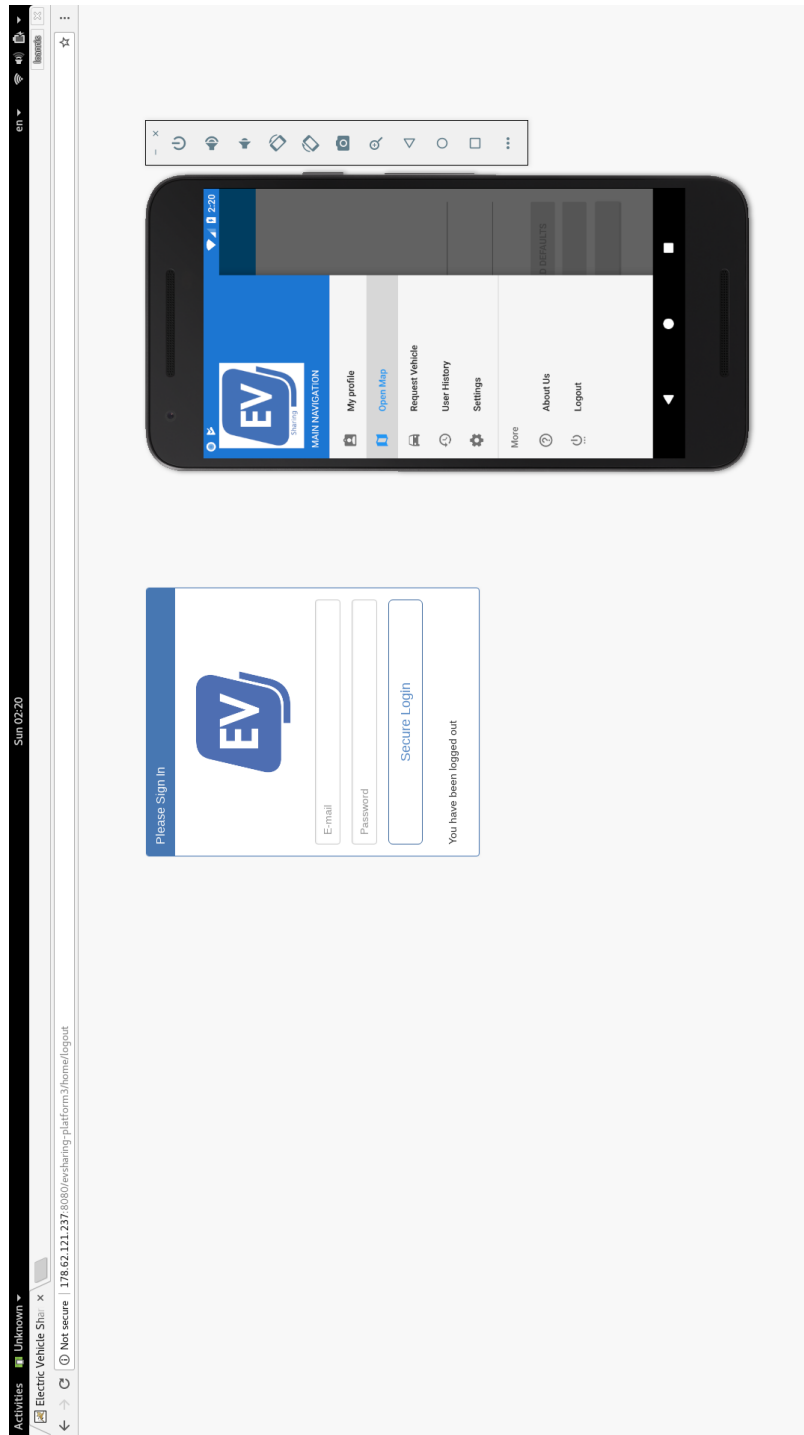


Figure 9.7.24: Navigation menu on the mobile application (part II).

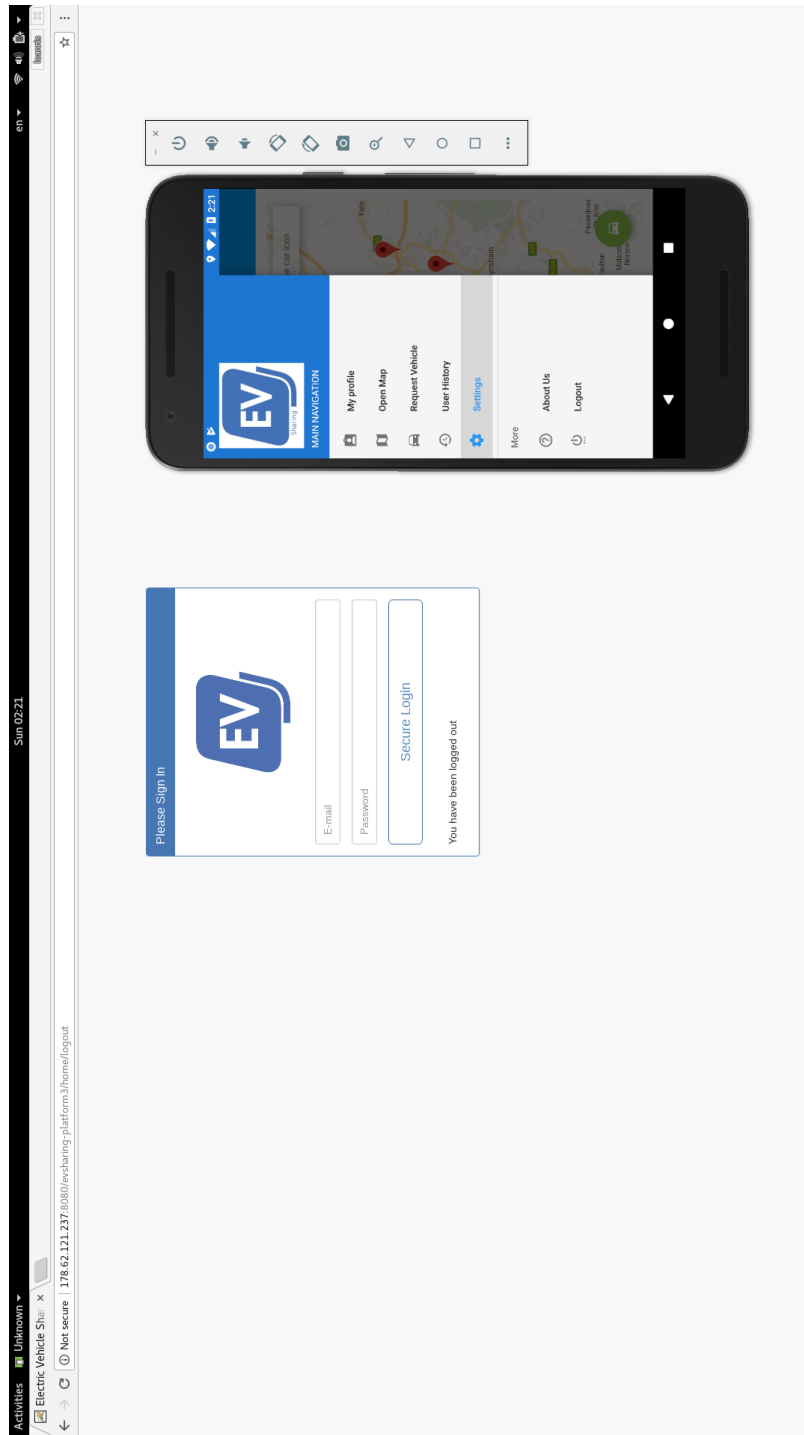


Figure 9.7.25: Navigation menu on the mobile application (part III).

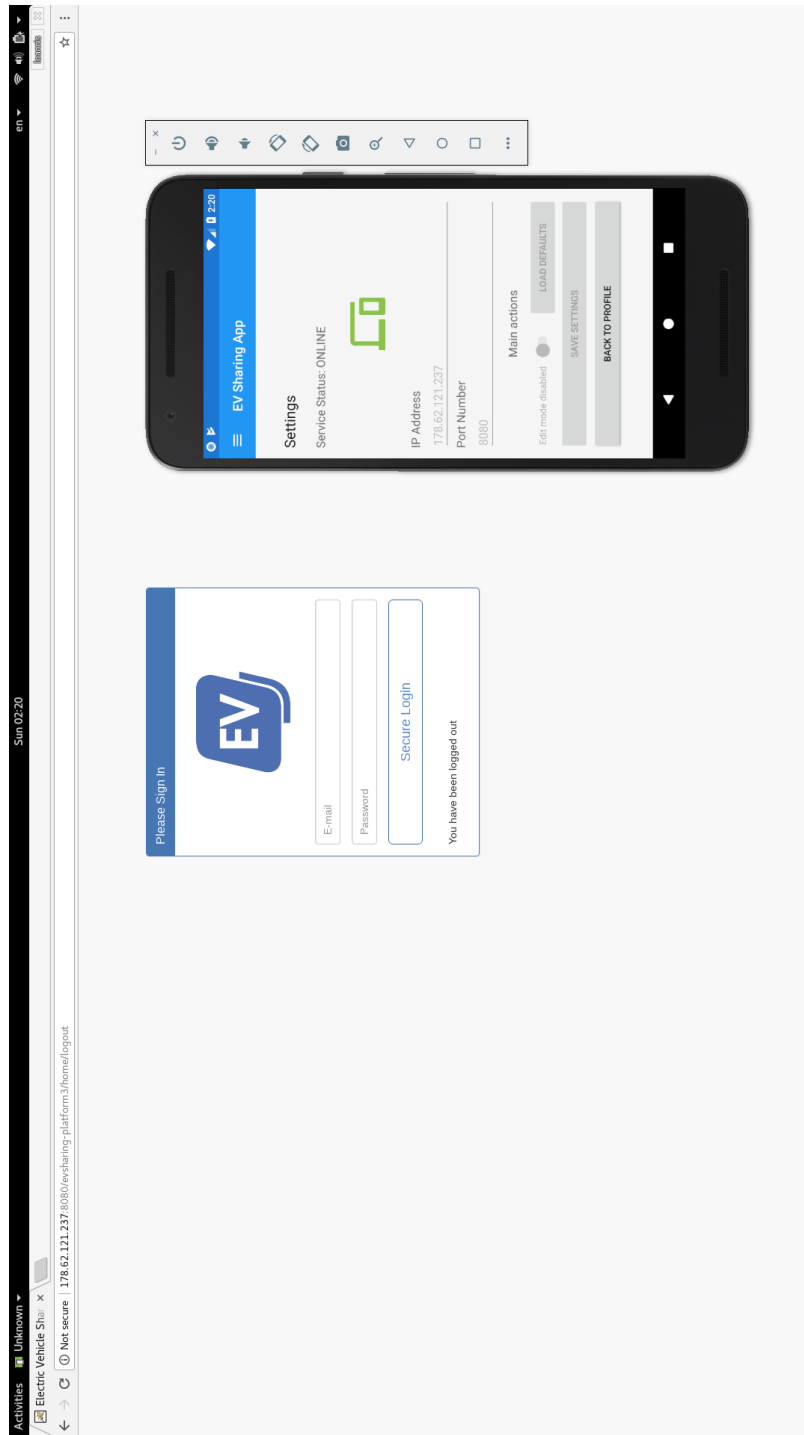


Figure 9.7.26: Settings screen on the mobile application.



Figure 9.8.1: Project presentation.

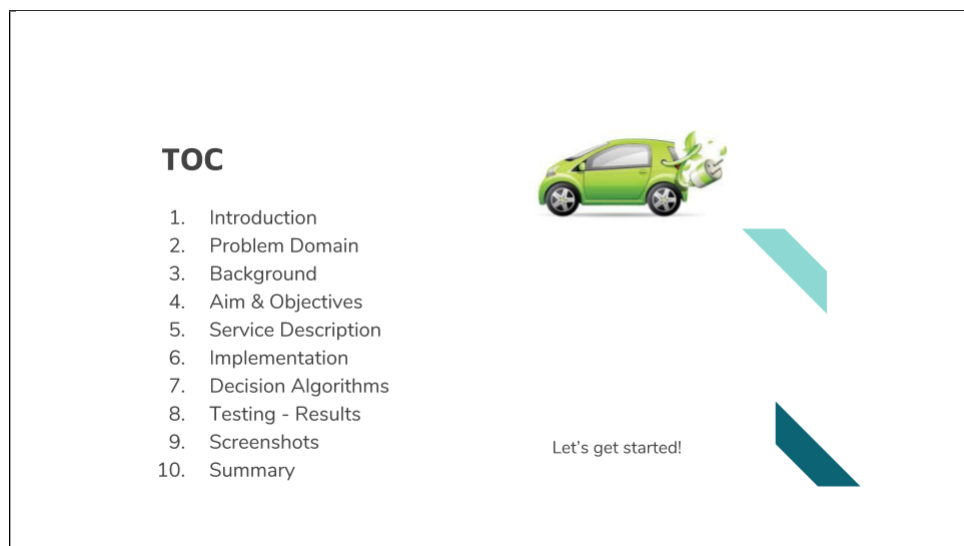


Figure 9.8.2: Table of contents slide.


1. Introduction

Increased motorization

- Urban environments as a pool of opportunities for our lives
- Transport is crucial for individual commuters and businesses
- More trips mean more ICE vehicles on the roads

So...

- ❖ How does this affect the environment?
- ❖ Should we worry about our health?



April 2018, ioanniegk@live.com

Figure 9.8.3: Introduction slide.

2. Problem Domain

Negative side effects (I)

- ICE vehicles produce air pollution
- GHG emission levels keep rising
- Congestion, traffic, accidents



Proposed solution

- ❖ Urban mobility Upgrade Part I: The EVs + Car Sharing
- ❖ Urban mobility Upgrade Part II: Software Platform for EVs



April 2018, ioanniegk@live.com

Figure 9.8.4: Problem domain slide.

3-1. Background

Negative side effects (II)

- Transport industry accountable for 74% of global CO₂ emissions (Rodrigue et al, 2006)
- Congestion harms the financial and social activities (Sperling et al, 2009)
- Road accidents will climb to 6th cause of death by 2020 (Gwilliam, 2002)
- ICE vehicles provide low fuel conversion rate of 30-35% (Mitchell et al, 2010)




Figure 9.8.5: Background slide (part I).

3-2. Background

The need for EVs

- ❖ EVs are eco friendly with very low harmful emissions and noise (Volkswagen Academy, 2013)
- ❖ EVs effectively utilize more than 90% of the supplied energy (Sperling et al, 2009)

Challenges for EVs

- Sparse charging infrastructure, "chicken and egg problem" (Burns et al 2002)
- A small BEV is still 30% more expensive than an equivalent ICE (Millard-Ball et al 2005)

April 2018, ioanniegk@lva.com

Figure 9.8.6: Background slide (part II).

3-3. Background

Car sharing systems

- ❖ Car sharing is the missing link to urban mobility (Millard-Ball et al, 2005)
- ❖ It is cost effective, allows residents to travel without owning a car (Bardhi et al, 2012)
- ❖ This leads to a 28% - 45% reduction in vehicle miles travelled and
- ❖ 19% - 54% lower GHG emissions for the average driver (Shaheen et al, 2007)

The change is happening

- ❖ The global threshold of 1 million EVs was surpassed in 2015 (Global EV Outlook 2016)
- ❖ Zipcar had more than 650,000 members and 8,900 cars in 2011 (Bardhi et al, 2012)

April 2018, ioanniegk@live.com

Figure 9.8.7: Background slide (part III).


3-4. Background



April 2018, ioanniegk@live.com

Figure 9.8.8: Background slide (part IV).

4-1. Aim & Objectives



A software solution to car sharing companies with EVs: Web Platform + Mobile App

- ❖ Manage user requests for hiring an EV to drive between two locations
- ❖ Ensuring that the highest number of people will be served in a given period


Objectives

- Administrators will be able to manage stations, vehicles, routes
- Users will be able to register, login and request a vehicle for a trip
- The system will handle user requests and decide if it will accept or deny them

April 2018, ioanniegk@live.com


Figure 9.8.9: Aim and objectives slide (part I).

4-2. Aim & Objectives



A software solution should be

- ❖ Reliable and easy to use
- ❖ Safe and secure
- ❖ Easy to install and configure
- ❖ Universal, suitable for car sharing companies
- ❖ Maintainable and extensible



April 2018, ioanniegk@live.com

Figure 9.8.10: Aim and objectives slide (part II).

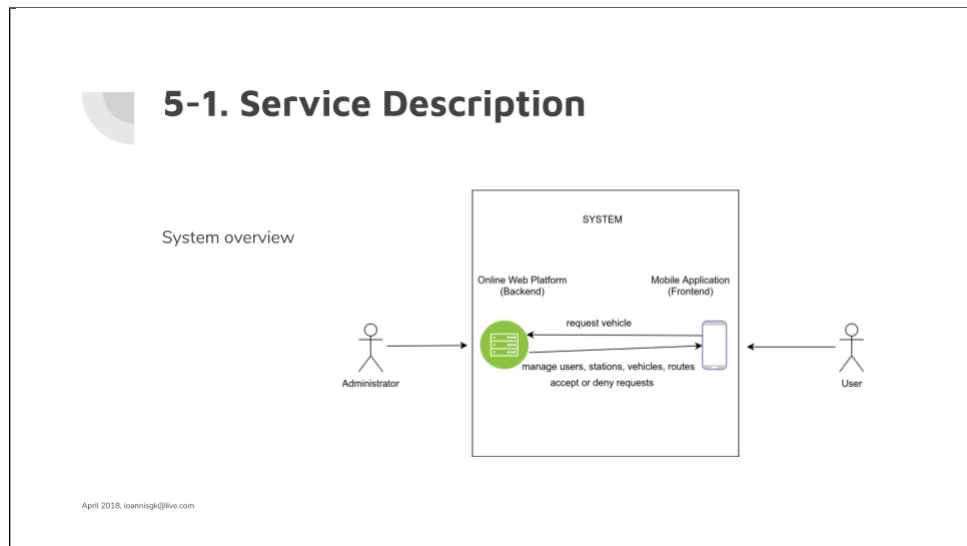


Figure 9.8.11: Service description slide (part I).

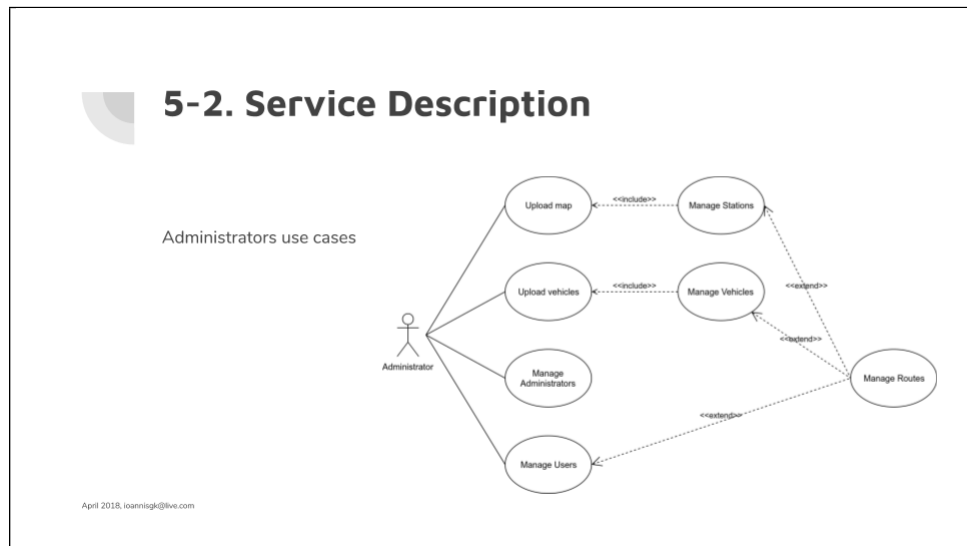


Figure 9.8.12: Service description slide (part II).

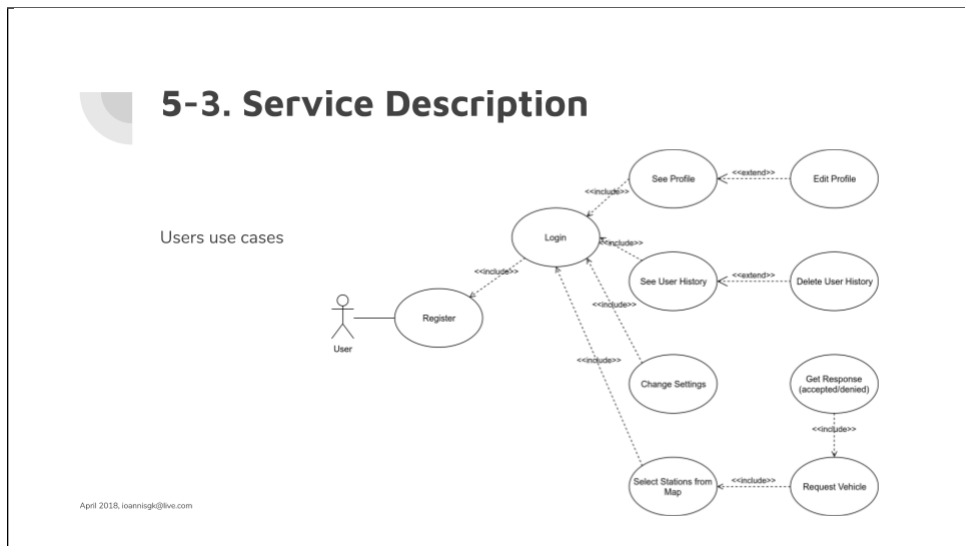


Figure 9.8.13: Service description slide (part III).

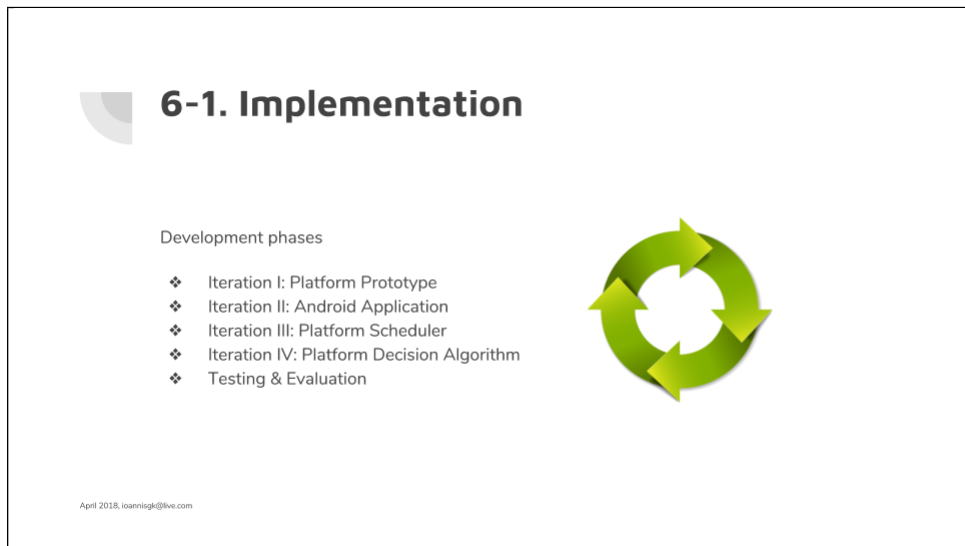



Figure 9.8.14: Implementation slide (part I).

6-2. Implementation



Spring Framework technologies

- Spring Core: flexibility with POJOs, Inversion of Control and Dependency Injection
- Spring MVC: web framework that follows the Model View Controller design pattern
- Spring Security: for authentication and authorization
- Spring Batch: for processing a high volume of batch operations
- Spring ORM: abstraction for database mapping APIs like Hibernate
- Spring AOP: to decouple code that implements functionality that should be separated

April 2018, ioannigk@live.com

Figure 9.8.15: Implementation slide (part II).

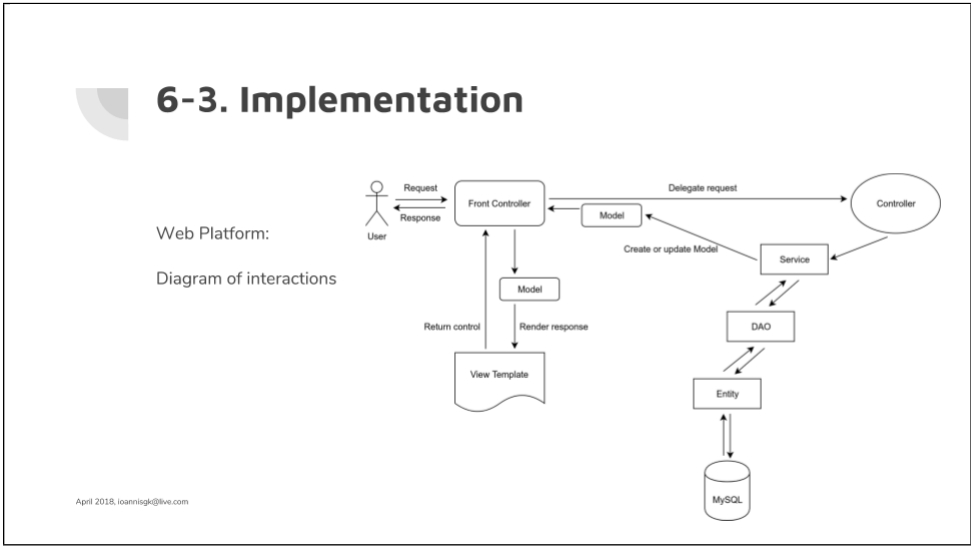


Figure 9.8.16: Implementation slide (part III).

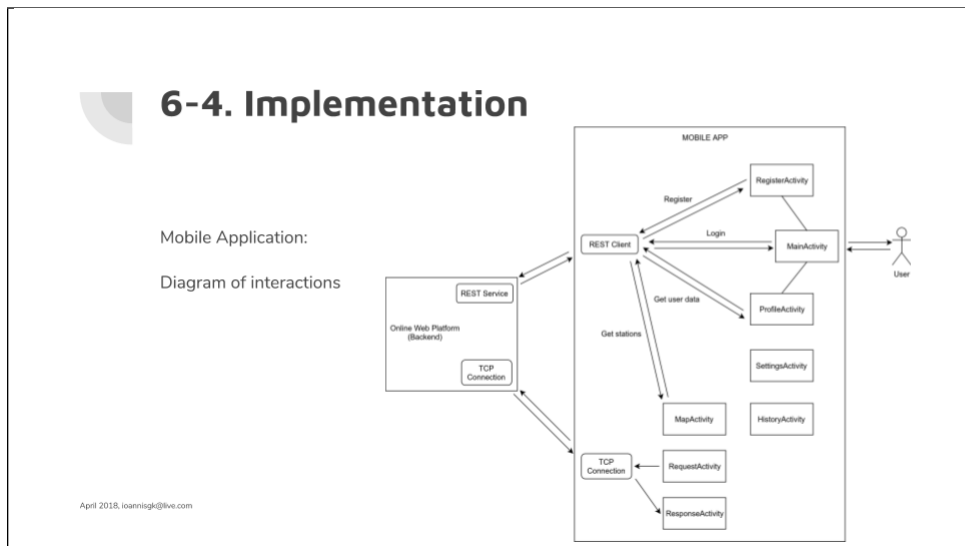


Figure 9.8.17: Implementation slide (part IV).

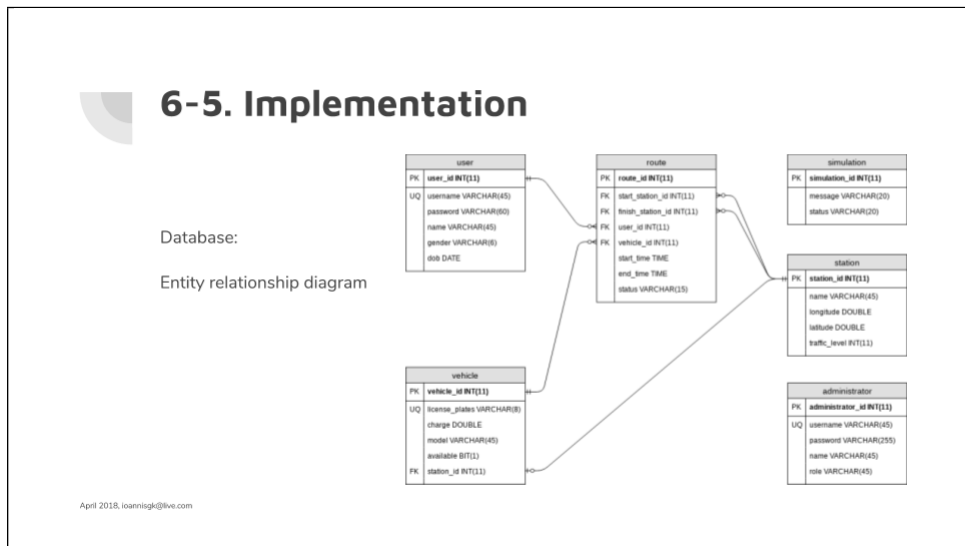



Figure 9.8.18: Implementation slide (part V).



7-1. Decision Algorithms


Short Mode:

- Fast execution
- Simple implementation
- Locked vehicles remain unavailable for hours

STEPS	SHORT MODE ALGORITHM
1	Extract message requests attributes
2	Get available vehicles in start station
3	Get available vehicles without future routes
4	Get available vehicles with enough charge
5	Assign best vehicle to new route
6	Save new route and result string

April 2018, ioannigk@live.com

Figure 9.8.19: Decision algorithms slide (part I).



7-2. Decision Algorithms

Long Mode:

- Fast execution
- Complex implementation
- It also considers the future routes of other vehicles in order to find substitutes and make current locked vehicles free for use

STEPS	LONG MODE ALGORITHM
1	Extract message requests attributes
2	Get available vehicles in start station
3	Get available vehicles without future routes
4	Get vehicles with one future route
5	Get available vehicles with enough charge
6	Assign best vehicle to new route
7	Save new route and result string
8	Else, get substitutes vehicles in start station
9	Get substitutes vehicles without future routes
10	Get substitutes vehicles with enough charge
11	Assign substitute vehicle to future route
12	Update future route with substitute vehicle
13	Assign best vehicle to new route
14	Save new route and result string

April 2018, ioannigk@live.com

Figure 9.8.20: Decision algorithms slide (part II).


8-1. Testing

Functional testing

- Component/Unit testing
- Integration testing
- System testing
- Acceptance testing

Test design techniques


- Black-box: equivalence partitioning, boundary value analysis
- White-box: statement and decision coverage measurement



April 2018, ioamiegk@live.com


Figure 9.8.21: Testing slide (part I).

8-2. Testing



Non-functional testing

- Security: test access to restricted content, web server security assessment
- Performance: test loading time, total page size and mobile app response
- Usability: heuristics for accessibility, identity, navigation and content



April 2018, ioamiegk@live.com

Figure 9.8.22: Testing slide (part II).

8-3. Testing

Case A: 60 requests for routes during a day

- Long mode performs better
- When time between requests increases, it performs even better
- Request allocation density does not affect system efficiency
- Average gain: 1.17%

April 2018, ioanmigk@live.com

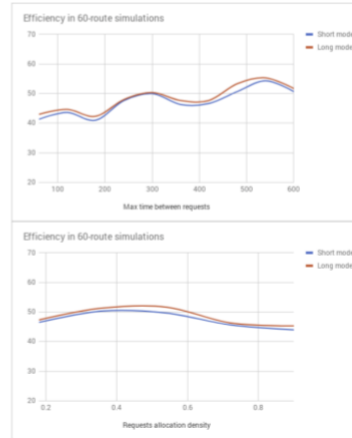


Figure 9.8.23: Testing slide (part III).

8-4. Testing

Case B: 120 requests for routes during a day

- Long mode performs better
- When time between requests increases, it performs even better
- When request allocation density increases, system efficiency drops
- Average gain: 1.83%

April 2018, ioanmigk@live.com

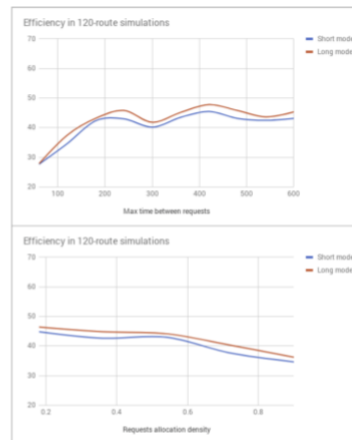


Figure 9.8.24: Testing slide (part IV).

9-1. Screenshots

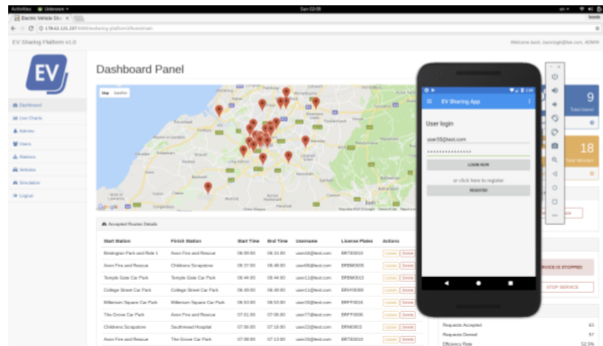


Figure 9.8.25: Screenshots slide (part I).

9-2. Screenshots

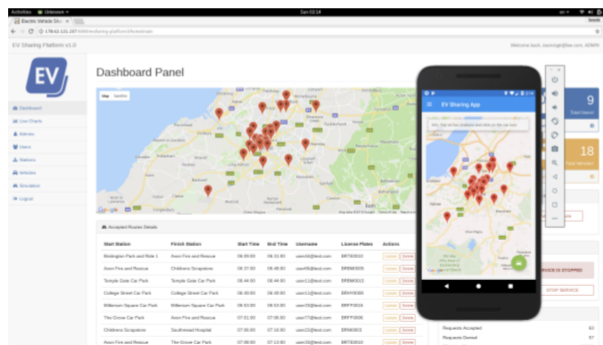


Figure 9.8.26: Screenshots slide (part II).

9-3. Screenshots

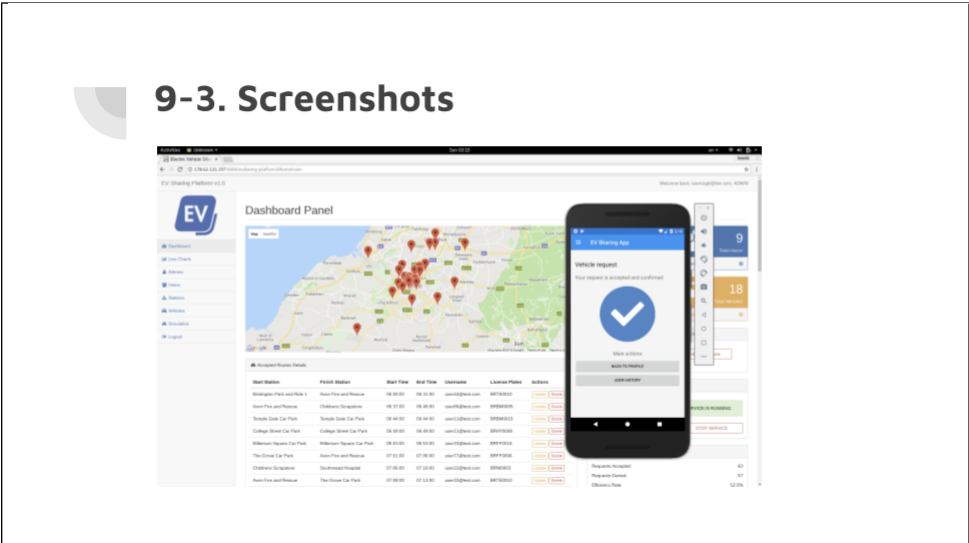


Figure 9.8.27: Screenshots slide (part III).

9-4. Screenshots

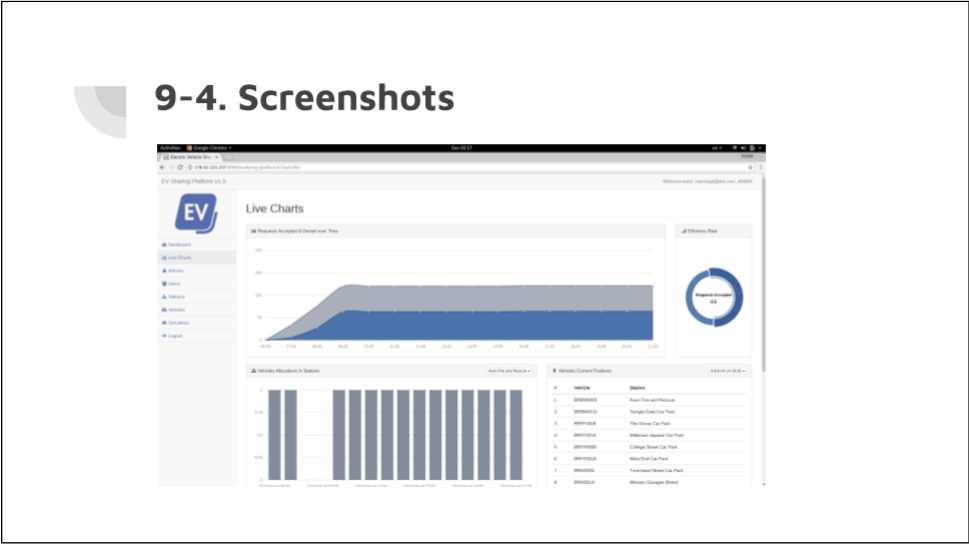


Figure 9.8.28: Screenshots slide (part IV).

9-5. Screenshots

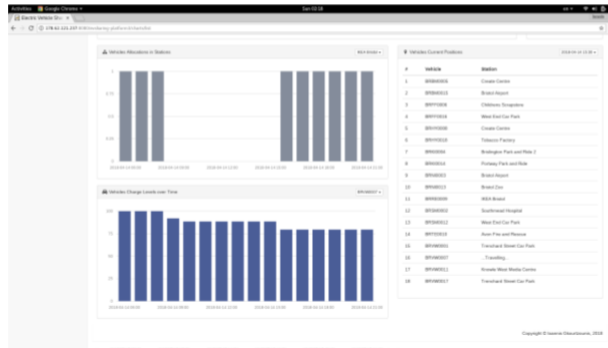


Figure 9.8.29: Screenshots slide (part V).

10.1 Summary




Our software solution: Web Platform + Mobile Application



- ❖ Reliable, easy to use by administrators and users, with great user experience
- ❖ Safe and secure, by encrypting messages between the platform and the mobile app
- ❖ Easy to install and configure with the use of XML files for maps & vehicles
- ❖ Universal, allowing effortless adoption by car sharing companies with EVs
- ❖ Maintainable and extensible, since it is developed with Spring MVC

April 2018, ioan.niegl@live.com

Figure 9.8.30: Summary slide (part I).




10.2. Summary - Links


- ❖ Video demo:
<https://youtu.be/flyixErIE-A>
- ❖ Source code:
<https://github.com/ioannisgk/evsharing-platform3-release>

April 2018, ioannisgk@live.com

Figure 9.8.31: Summary, links slide (part II).



References



Bardhi, F. and Eckhardt, G. (2012). Access-Based Consumption: The Case of CarSharing. *Journal of Consumer Research*, 39(4), pp.881-898.

Burns, L. D., Mc, J. B., Jablonski, N. G., Chaplin, G., Casagrande, R., and Voldman, S. H. (2002). Vehicle of Change. *Scientific American*, 287(4), pp.64-73.

Global EV Outlook (2016). *Beyond One Million Electric Cars*. Paris: International Energy Agency.

Gwilliam, K. (2002). *Cities on the move: a World Bank urban transport strategy review*. Washington, DC: World Bank.

Millard-Ball, A., Murray, G., Schure, J., Fox, C. and Burkhardt, J. (2005). *Car-sharing: Where and how it succeeds*(Vol. 108). Washington, D.C: Transportation Research Board.

Mitchell, W., Borroni-Bird, C. and Burns, L. (2010). *Reinventing the Automobile*. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Rodrigue, J., Comtois, C. and Slack, B. (2006). *The Geography of Transport Systems*. London: Routledge.

Sperling, D. and Gordon, D. (2009). *Two Billion Cars: Driving toward sustainability*. Oxford: Oxford University Press.

Volkswagen Academy (2013). *Basics of Electric Vehicles*. Volkswagen Group.

Figure 9.8.32: References slide.

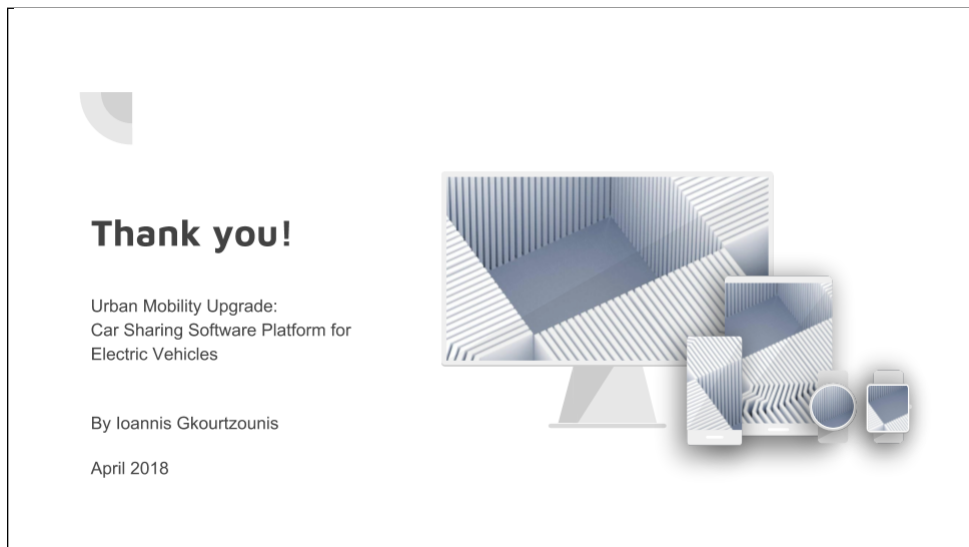


Figure 9.8.33: End of presentation.

Colophon

THIS DISSERTATION was submitted to the University of Northampton in partial fulfillment of the requirements for the degree of Bsc(Hons) Computing by Ioannis Gkourtzounis in April 2018. A video demonstration is available at (live link) youtu.be/flyixErIE-A. All project files can be found online at github.com/ioannisgk or from the author at ioannisgk@live.com.