Politecnico di Milano A.A 2016/2017

Software Engineering 2 project:

PowerEnJoy

Requirement Analysis and Specification Document

(RASD)

Alessandro Perini, Federico Saini, Ali Merd Türkçapar

Version: 1.0 Release date: 13/11/2016

Contents

Chapter 1
Introduction

1.1 Purpose

This document represents the Requirement Analysis and Specification Document (RASD). The main purpose is to provide a complete and clear representation of the reality that the system describes. This representation is important both for the commissioner of the software and for the developers. It is useful for the commissioner to have a clear idea of the necessities he needs and the functionality that the system will provide. It is also an important landmark for the developers that will refer to all the specifications described in it.

1.2 Scope

What we want to create is a new car sharing service, called PowerEnjoy, based exclusively on ecological electrical vehicles.

Users will be able to search for a car through a web application or a mobile application. It is possible to reserve a car located inside the service area of the system and pick-it up later. So the main scope of the project is to allow people to use a car for short rides around the city.

We want to incentivize positive and altruistic behaviours. To make it possible, users can get a discount on their rent fees if they share the ride with at least two other passengers and if they park the car in a power grid station or with an adequate level of battery. At the end of the ride users will be charged according to a function of the time they spent using the car and the cost is modified with a discount described above or with an extra-fee in case of a bad behaviour generating the total cost.

Goals

Below, a list of the goals that the system should achieve. All the related requirements will be listed in the chapter three of this document.

[G1] Every visitor must be able to register to the system.

[G2] A non logged user must always be able to login.

[G3] A user must be able to search for an available car according to his position or a given address.

[G4] A user can reserve an available car and the reservation stands for one hour.

[G5] A user that reaches the car he reserved must be able to unlock it.

[G6] The user must be aware of the cost of the trip during the race.

[G7] A user can always end the race inside the service area.

[G8] A race with at least 3 people in the car should get a discount of 10% of the total price.

[G9] Based on the battery level and the park position, some extra fees or some discounts can be applied to the cost of the race.

[G10] A user can get a discount if he enables the money saving option.

[G11] A user should be able to report an issue that will be solved by technicians.

1.3 Definition, abbreviation

Definition:

Visitor: a guest that is not registered into the system.

User: a person that already made the registration.

Credentials: a set of user’s information (name, surname, email, driving license).

Payment information: all information of a valid credit card (owner, number, security code, expiration date).

Passenger: a person that travels in the car during a race that is not the user who drives.

Race: a trip that a user makes from when he unlocks the car since the ending procedure ends.

Issue flag: a boolean value that indicates the presence of any problem on the car (broken engine, punctured wheel, scratches, etc..).

Status: information about the car (position, battery level, CarID, issue flag).

Power Grid Station: parking spot where there are power plugs to charge the car's battery.

Safe Area: a circle with a radius of 3km from a power grid station.

Service Area: it’s the set of zones, defined in the Service Map, where user can park the car.

Service Map: a map that shows all the parking zones, power grid stations and safe areas.

CarID: is the car licence plate. This is used by the system as a unique identifier for the cars.

Web-app: web service available through a web browser.

Mobile-app: application installed on a mobile device.

Counter: a counter that takes count of the actual cost of the race.

Timer: a timer that counts the minutes passed from a car reservation.

Unlock button: a virtual button shown on the mobile-app after a car reservation.

Money Saving Option: an option that indicates the best station to park the car based on the result of the money saving option algorithm.

Ending Procedure: a set of operations that has to be performed, before ending the race, to allow the system to change the car state from In-use to Available or Maintenance.

Issue solved button: virtual button present on the mobile-app, when pressed it changes the issue flag to false and it locks the selected car.

Report: all the information about a reported issue (status of the car, sentence from the user, type of issue, user who reports the issue, timestamp)

OnBoard Unit: software module embedded on the car.

States of a car:

Available: a car that can be reserved, with issue flag equal to false and with at least 40% of battery.

Reserved: a car reserved by a user.

In-use: a car that is being used by a user.

Maintenance: a parked car with battery level below 40% or with issue flag equal to true.

Abbreviation:

MSO: Money Saving Option.

PGS: Power Grid Station.

Rn: Functional requirement.

Nn: Non functional requirement.

Dn: Domain assumption.

Pn: Performance requirement.

Gn: Goal.

1.4 Reference documents

[1] *IEEE Std 830-1998 "IEEE Recommended Practice for Software Requirements Specifications"*

[2] *http://alloy.mit.edu/alloy/*

[3] PowerEnJoy specification document: "ASSIGNMENTS AA 2016-2017.PDF".

1.5 Overview

The rest of this document is organized in 4 chapters.

- Chapter 2 (Overall Description) provides an overall description of the product and its requirements. This chapter does not state the requirements but provides a background for those requirements, which are defined in detail in the following chapter. The purpose of the overall description is to make requirements easier to understand.

- Chapter 3 (Specific Requirements) contains all the software requirements. This is the most important part of the document because contains the detailed description of all the functionalities that the system will provide.

- Chapter 4 (Alloy Modeling) contains a representation of the system made with Alloy language [2]. Using the Alloy Analyzer it's possible to find all the configurations that satisfy the constraints. So the purpose of this chapter is to find any possible missing or wrong constraints.

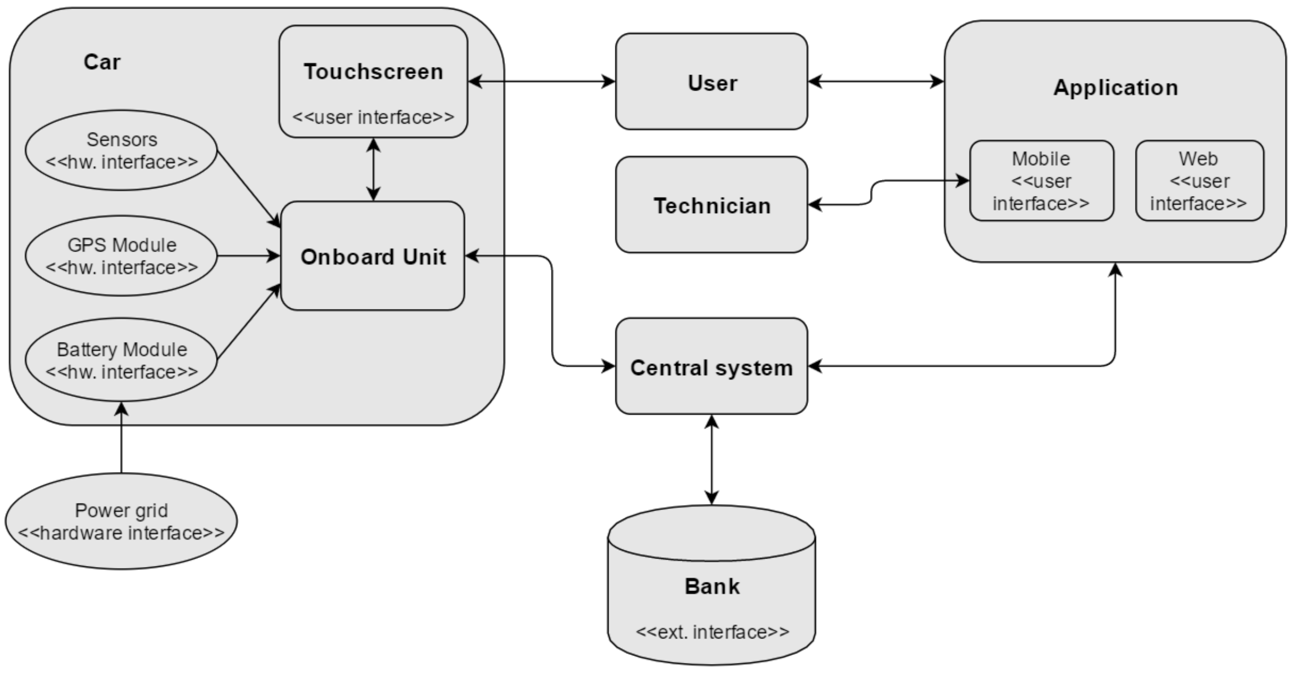
- Chapter 5 (Appendixes) contains other information like: hours of work by the member of the group and all the revisions of the document.

Chapter 2
Overall Description

2.1 Product perspective

The PowerEnJoy car sharing service is a self-standing system containing different parts: cars, a centralized software, web and mobile apps and a maintenance service. The target is the development of a digital management system, that will integrate the physical devices (cars, charging stations), users and the maintenance service. So the software that will be produced, will work only with the other parts of the system.

Diagram of the system



2.2 User characteristics

The system is intended for people who travel around the city and want to use the car to make short race in it. People interacting with the system, are categorized as visitors or users. Visitor, as described in the glossary, are people that are not registered into the system and are allowed only to use the registration function. Users, on the other hand are the people that are registered into the system.

A visitor, needs to subscribe the Terms and Conditions document to become a user. Except that document, user is not required to have other technical knowledge to use the system.

2.3 Constraints

No constraints on the system to be developed were given by the customers.

2.4 Assumptions and Dependencies

2.4.1 Car

- Sensors located on the car always work properly.
- Below 5% of the battery level, the engine is unable to work but the car system still works until the complete exhaustion of the battery charge.

2.4.2 GPS

- The position provided by the car's and mobile-app's GPS module is correct.
- The car GPS module always sends his position while the battery is not exhaust so the system can always know the position of a car.

2.4.3 Technician

- At any time during the day, at least one technician is working in the service area.
- He is able to recharge cars on-site wherever they are.
- He is able to fix any problem on the car.

2.4.4 Users

- Users have read and understand the Terms and Conditions document.
- Users must ensure a proper use of the system, described in the Terms and Conditions document.
- The driving licence inserted in the system by a user during the registration must be valid during any race.
- Bank payment interface never refuses a transaction.

2.5 Decisions

2.5.1 Issues

Issue are divided in two categories: soft issues and hard issues. A soft issue is a problem that not necessarily needs a technician intervention (scratches, dents, ruined seats, etc..). For these issues, the car in maintenance is switched to available automatically by the system and the issue is saved. For the hard issues (punctured wheels, broken windscreens, problem on the engine, etc..), the car in maintenance needs a technician intervention. All the reported issues, will change the state of the car to maintenance. We use this policy to allow the customer to have more flexibility on decisions. For example, if he decides to change all the ruined seats, he can tell the system to not automatically change to available a car with a soft issue without changing the steps between the states. So all the issues can be moved from a category to the other.

2.5.2 Discounts

Summary table of discounts and extra fees:

Discount / Extra fee	Reason	Percentage on the race cost
Discount	Passenger > 1	10%
Discount	Plugged	30%
Discount	Battery level >= 50%	20%
Extra fee	Outside Safe Area	30%
Extra fee	Battery level <= 20%	30%

The system applies at most one discount and one extra fee to the cost of the race considering the higher discount. In this way, a user will never have more than 30% of extra fee and 30% of discount.

2.5.3 Battery

We decide that a car left with less than 40% of the battery full must change its state to maintenance. To return available, the car must be plugged to a power support by a technician and battery must rise up to 40%. This choice is made to ensure that any car that a user can reserve will have at least 40% of battery. Also because of the extra fee due to the end of a race with low battery (20%). With this choice a user has at least 20% of battery to use before incursion of an extra fee.

When the battery level reaches 5% during a race, the car will stop and will allow the user to stop the race. We assumed that a 5% is a critical level. This means that the engine is unable to continue to work but the car system still work and the race can be ended properly.

2.5.4 MSO Algorithm

The MSO algorithm will show to the user the best charging station where park the car. It must be programmed to respect the following points:

- The best station is chosen from a group that is composed of considering all the charging stations with at least 1 available plug, at most 1,5 km far from the destination address.
- Among the group, the algorithm will select the one with more free plugs.
- If there are no stations in the group, the algorithm will select the charging station with at least one available plug closer to the destination address.

We considered a circle of 1,5 km radius from the destination address to enable the user to reach it by walk after he parks the car. The algorithm will select the charging station with more free plugs in order to satisfy the uniform distribution of cars.

This algorithm presents also an advantage for the user: it is the only way to know for sure the location of a parking spot where to get a 30% discount.

Once activated, the MSO doesn’t force the user to plug the car at a power grid station: if at the end of the race the car is not plugged the 30% discount will not be considered. It is also possible that the user parks the car in power grid station that is different from the one indicated by the system, in this case the user still gets the 30% discount.

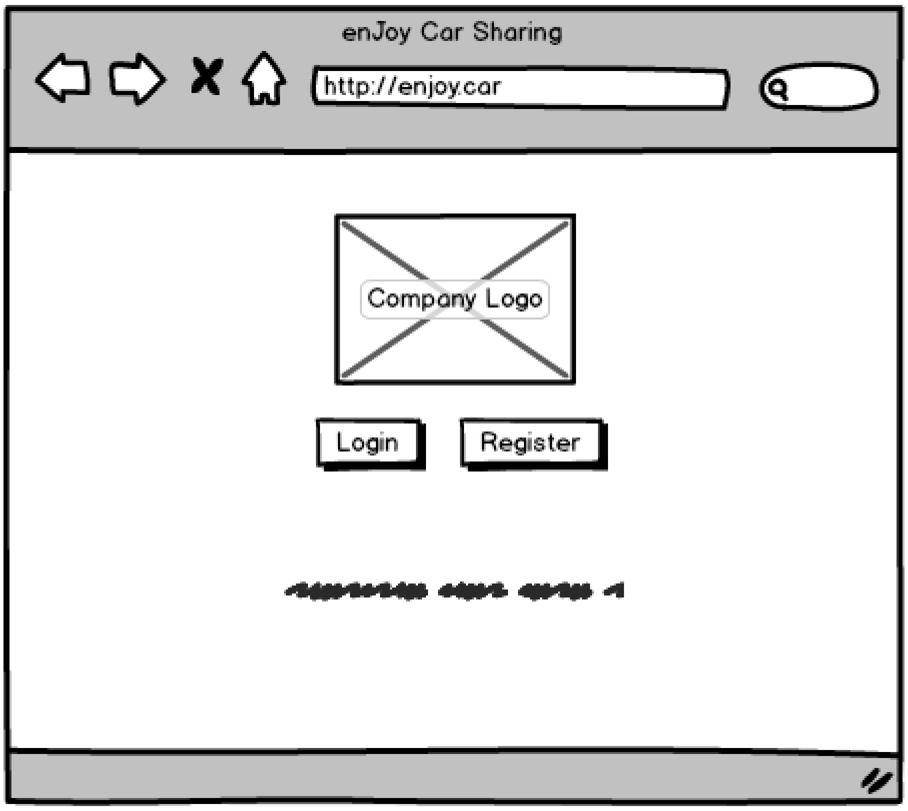
Chapter 3
Specific Requirements

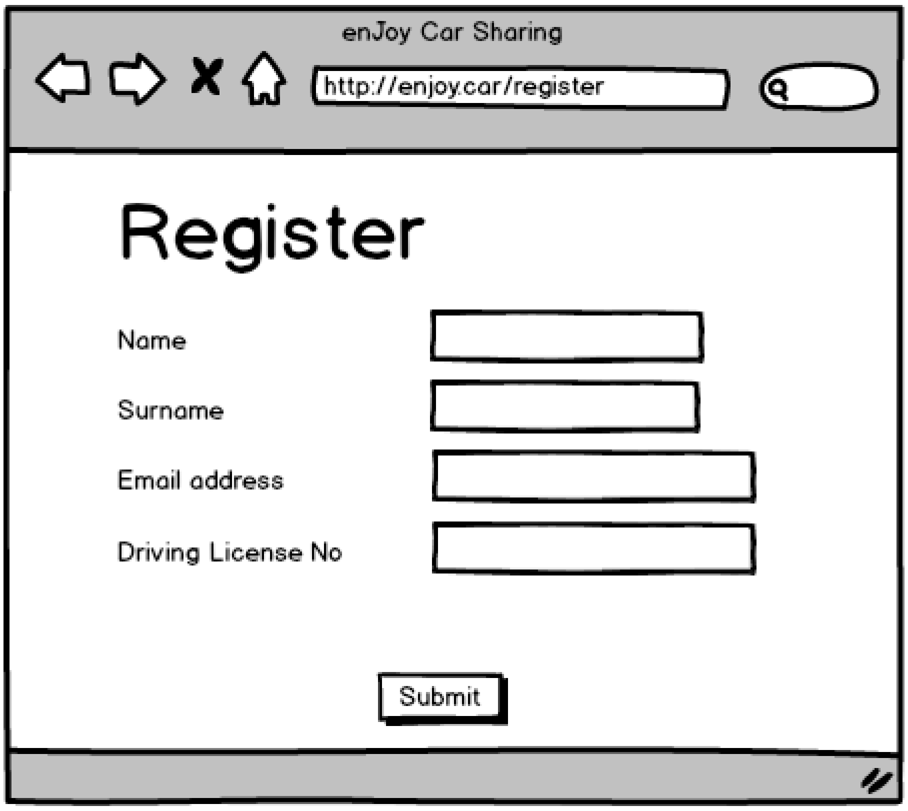
3.1 Interface Requirements

3.1.1 User Interfaces

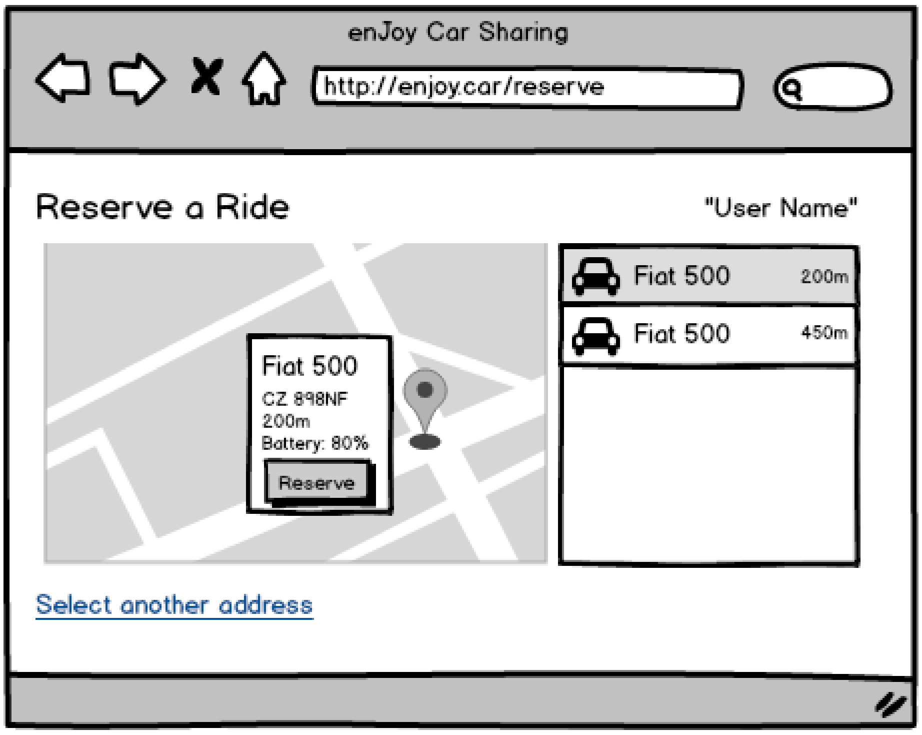
Web-app

This is a sample of the website welcome page. From here the user can log into the system using the related button. This page also gives the opportunity for a visitor to register himself into the system using the register button. The register button will open a new page with the form to be filled for complete the registration.



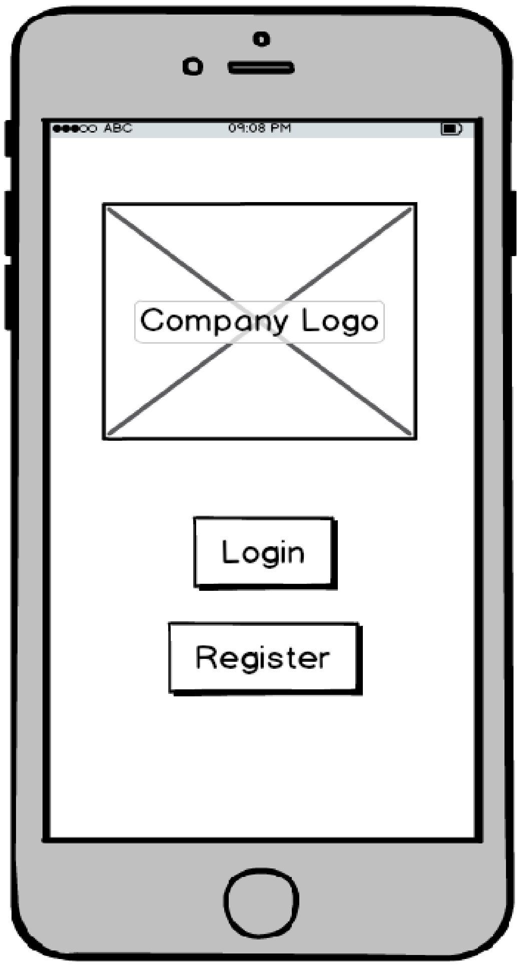


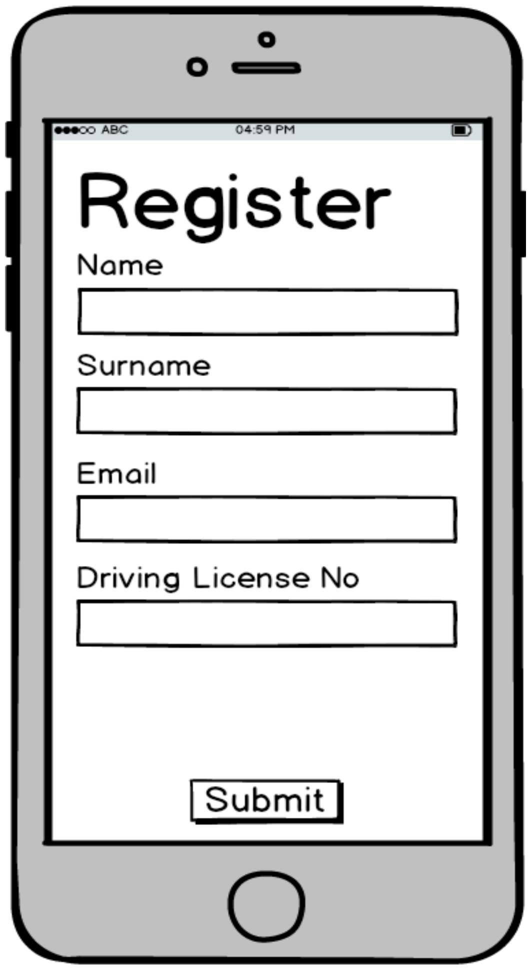
Below is showed the reservation screen. User can see a list of available cars with all the related information. User can choose to see available car near his position or near a given address.

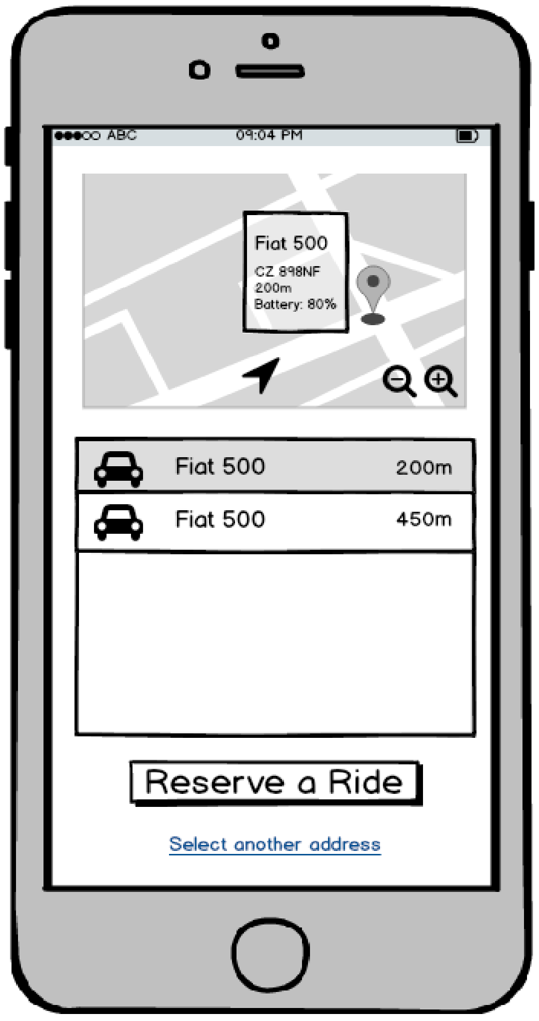


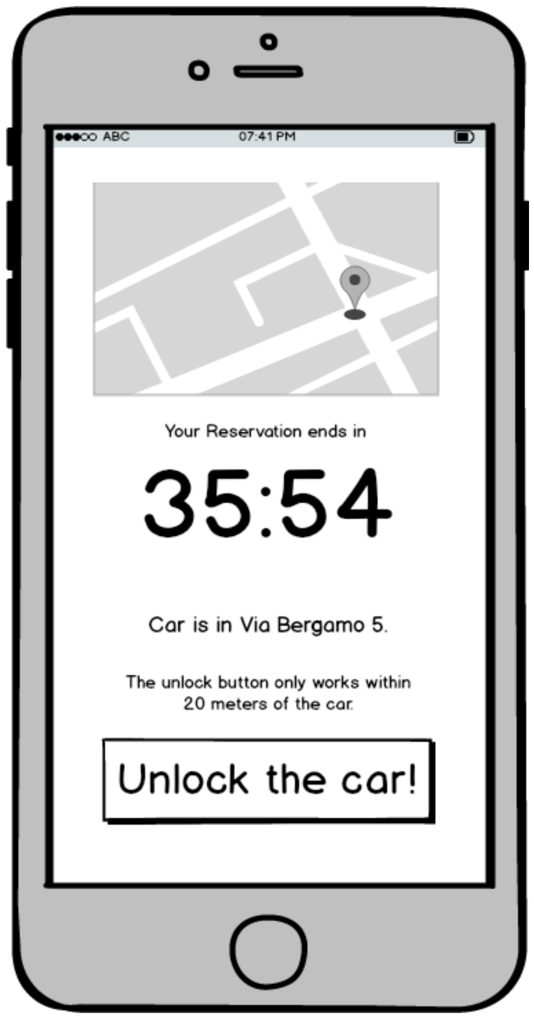
Mobile-app

Below are shown all the mobile counterpart pages for registration and login.









Car is in Via Bergamo 5.

The unlock button only works within 20 meters of the car.

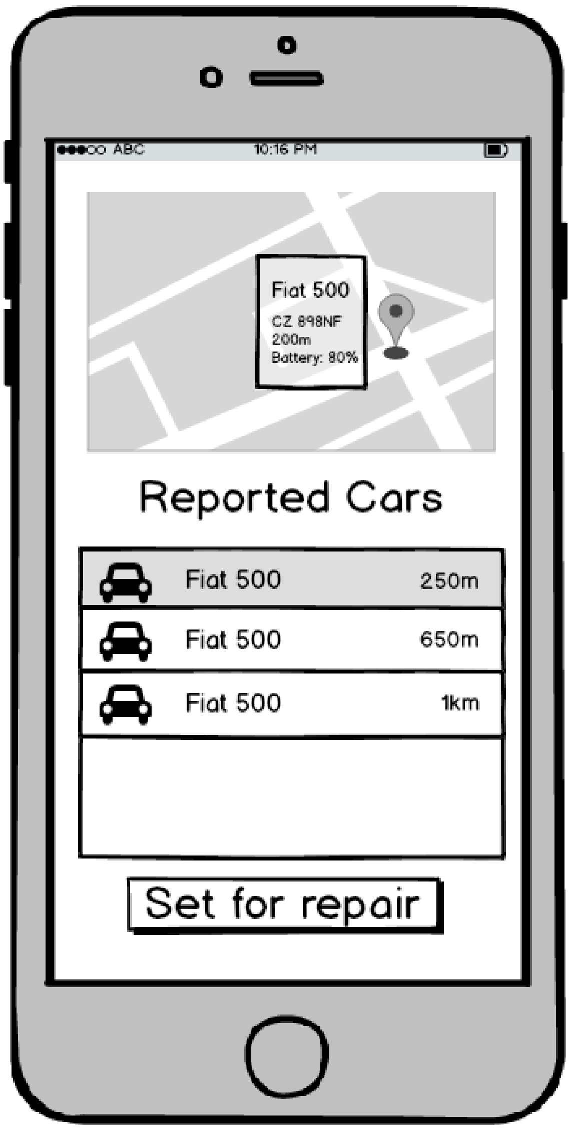
Reservation and unlock the car pages. The button to unlock the car is showed as soon as the car is reserved but it effectively unlocks the car only when the user is at most 20 meters far from it. User can see his position with the black arrow on the map on the reservation screen.

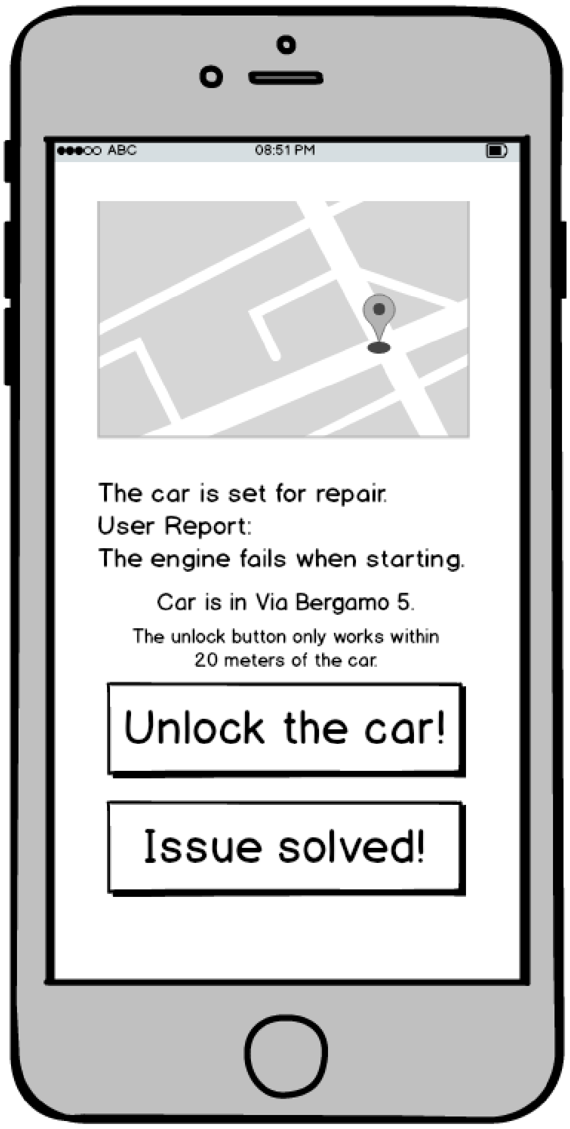
Technician UI

The technician section of the mobile-app shows the reported cars and technician can set a car for repair then read the user report about it.

Below in the left you can see the screen for the technician to see the reported cars, similar to the reservation screen the technician can see the cars with respect to his location.

When the user selects a car, it is shown in the map and the technician is able to unlock the car when he is within 20 meters of the car. After fixing the issue he can click the “issue solved” button and it will remove the car’s issue flag.





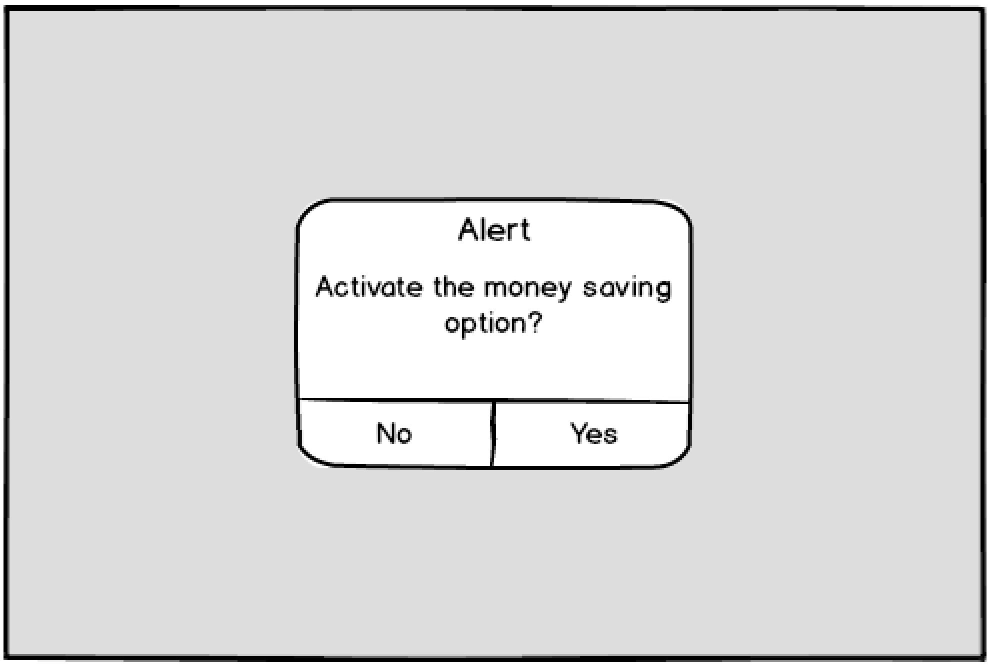
The car is set for repair.
User Report:
The engine fails when starting.

Car is in Via Bergamo 5.

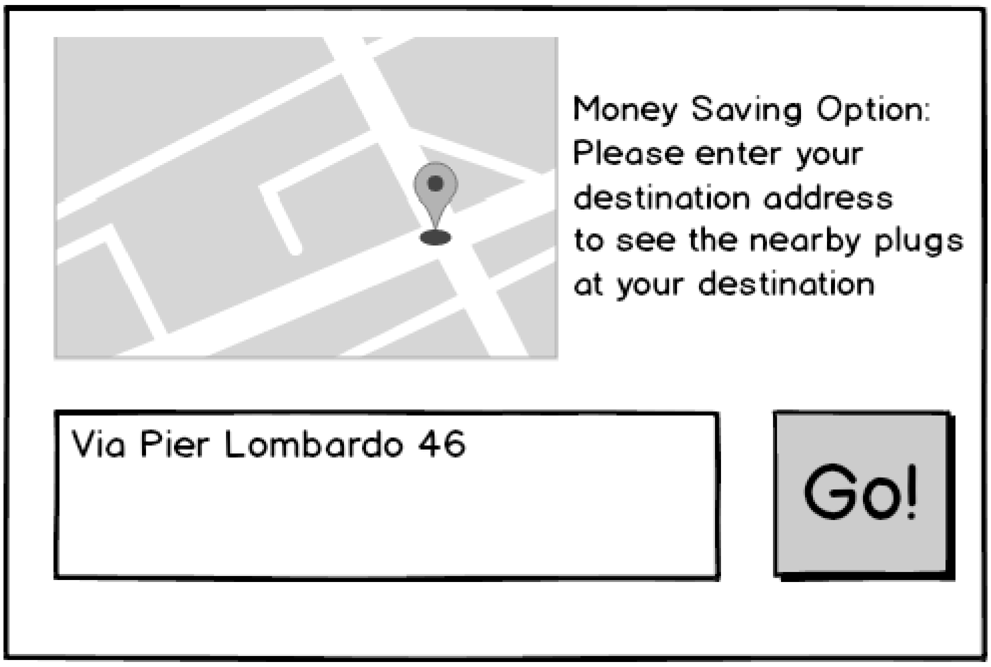
The unlock button only works within 20 meters of the car.

Touchscreen

This is the user interface for the touchscreen in the car. Below is showed the welcome screen of the car application asking for the activation of the MSO. If the user select “No” it will be shown the trip screen.

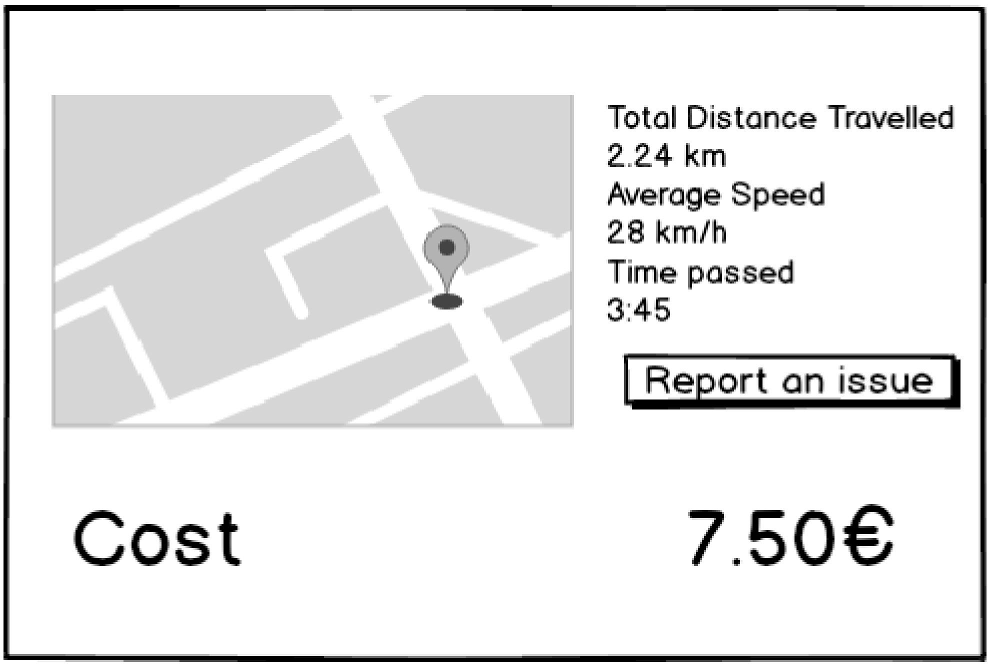


If the user taps “Yes”, the application will ask the user to enter the destination address. Based on the destination, the system shows the best plug. Then, the trip screen is showed.


Money Saving Option:
Please enter your
destination address
to see the nearby plugs
at your destination

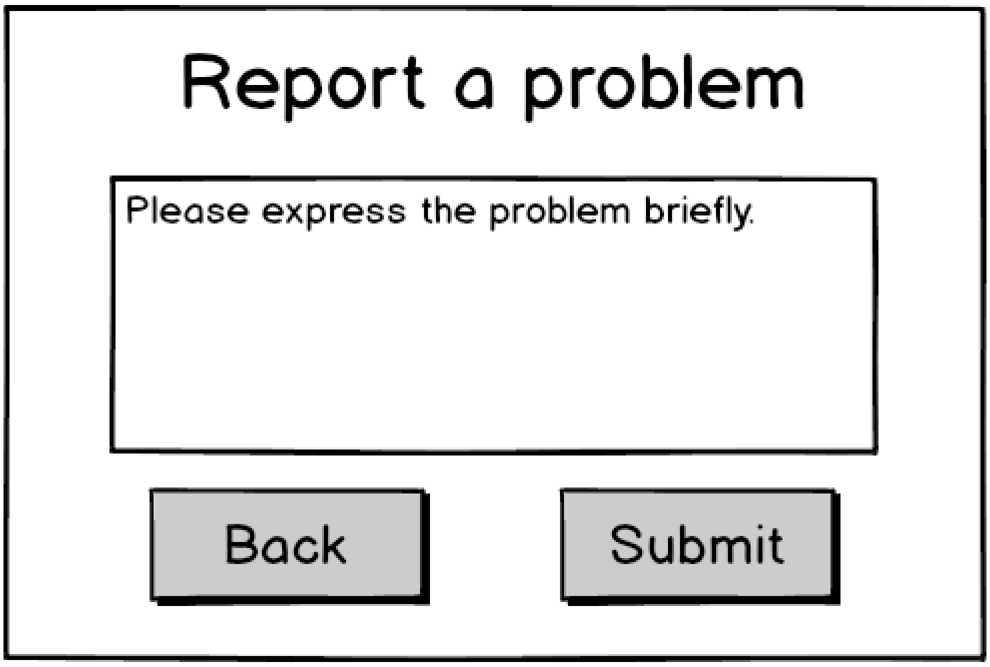
Via Pier Lombardo 46

The trip screen shows the cost of the trip, the total distance travelled, average speed and total time spent from the start. User also has the option to report an issue from this screen.

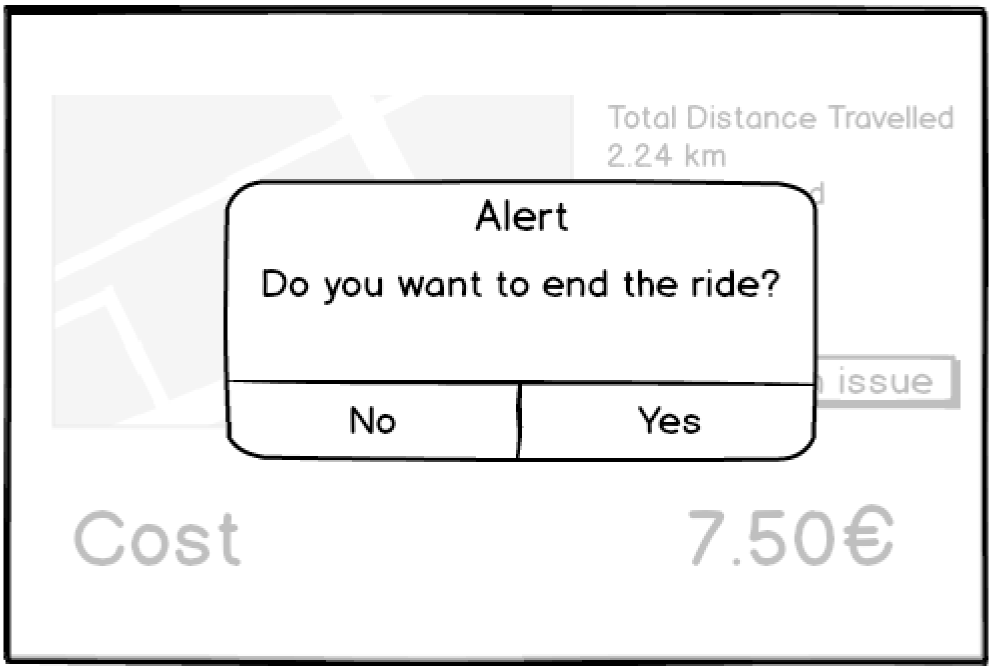

Total Distance Travelled
2.24 km
Average Speed
28 km/h
Time passed
3:45

Cost 7.50€

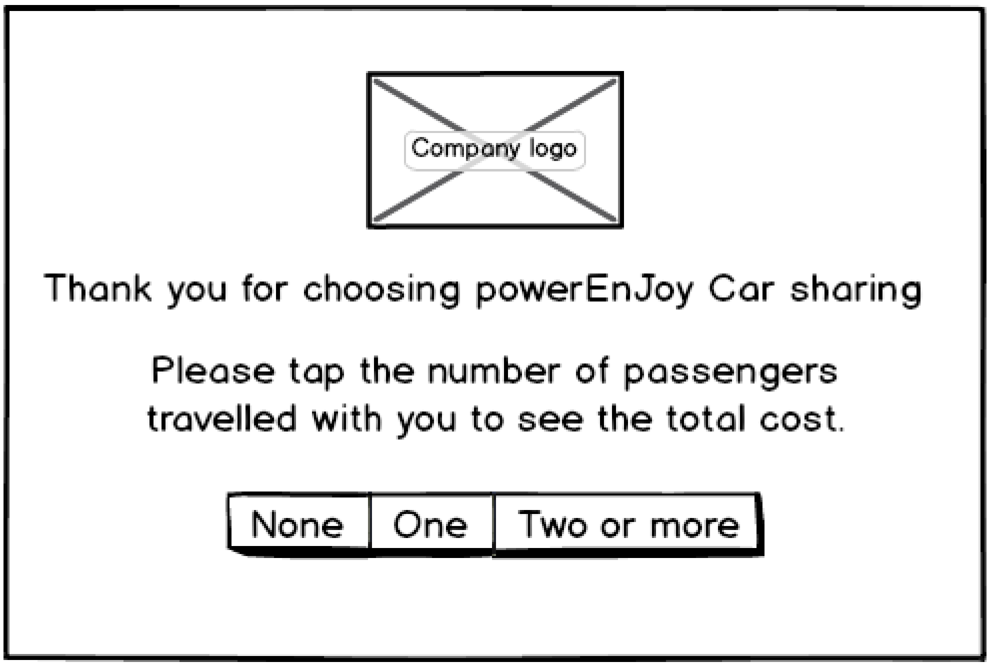
When the user taps the “Report an Issue” button he has to select the type of issue. Then the user can write a brief message and then submit the report. If the user taps “Back” the issue is not reported and the trip screen continue.



Every time the engine is powered off, the following screen is showed. Tapping “Yes” the ending procedure starts. The same screen it is also shown when an issue is reported.



If the user clicks “Yes” below screen appears asking the user number of passengers in the car to apply a discount if the number of passengers is two or more. Then the total cost of the ride is shown to the user.


Company logo
Thank you for choosing powerEnJoy Car sharing
Please tap the number of passengers travelled with you to see the total cost.
None
One
Two or more

3.1.2 Hardware Interfaces

Power Grid Station

The power grid stations are a set of available electric plugs located in the middle of a safe area where the user can park the car. Every station sends in real time data about the total number of plugs present in each station and the number of plugs that are not in use.

Sensors

Every car is equipped with 5 binary sensors that continuously send a signal that can be interpreted in this way:

- Sensors 1-4
  - S_n = 1 means that the door number n is opened, otherwise it is closed.
- Sensor 5
  - S_5 = 1 means that the car is actually plugged into a power grid station.

Batteries

The battery present on the car continuously sends to the OnBoard Unit its level of charge expressed by an integer number between 0 and 100. This number represents the percentage of the battery's charge that can be used.

The battery is also able to recharge itself when the car is plugged to a PWS. Once the car is plugged, the battery immediately starts to recharge until it is fully charged.

3.1.3 Software Interfaces

Bank Payment Interface

In order to bill the user and to ensure the security of the cars, our system has a bank payment interface that is able to provide the following features:

- it can transfer money from user's account to the PowerEnJoy bank account using the user's credit card information.
- Can check in real time the validity of a credit card during the registration phase.
- As our system is available 24h/7, it is the same for the bank payment interface.

3.1.4 Communication Interface

Our cars are equipped with a system able to transfer information with a mobile internet access. We need to communicate with cars regardless their position. In case of missing signal, we assume that the position of car is the last position the car sent.

3.2 Functional Requirements

Below the list of functional requirements related to the goals.

[G1] Every visitor must be able to register to the system.

R1: A visitor, to register into the system, must provide valid credentials and payment information.

R2: The system must be able to check the validity of the visitor credentials and payment information.

R3: If the information received are correct, the system must reply to the visitor with an email containing a random generated password.

R4: Registration is the only function that a visitor is allowed to use.

[G2] A non logged user must always be able to login.

R5: The system must always provide the login function to a not already logged user.

R6: Only users can log into the system.

R7: A user can log into the system only if the email and password indicated are correct.

R8: All the functionality of the system, except for registration and login, are provided only to logged users.

R9: System must avoid multiple simultaneous login from a user.

[G3] A user must be able to search for an available car according to his position or a given address.

R10: The system should allow the user to search for an available car sorted by the distance from his position.

R11: The system should allow the user to search for an available car sorted by the distance from an indicated address.

[G4] A user can reserve an available car and the reservation stands for one hour.

R12: The system should allow user to reserve only Available cars.

R13: A user can reserve only one car at a time.

R14: Only a user can reserve a car.

R15: If the user doesn’t unlock the car after one hour from the reservation, the car returns Available again.

R16: If the user doesn’t unlock the car after one hour from the reservation, the system charges the user with 1€.

R17: The reservation must hold even if the battery level goes below 40%.

R18: User cannot cancel a reservation without reporting an issue.

[G5] A user that reaches the car he reserved must be able to unlock it.

R19: Reserved cars can be unlocked only by the user that made the reservation.

R20: A user can unlock only Reserved cars.

R21: The user should be able to indicate to the system that he reaches the reserved car and he wants to unlock it.

R22: The system should unlock the reserved car only if the user asked for it.

R23: The system should unlock the reserved car only if the user is at most 20 meters far from it.

R24: After the user unlock the car, the car state will change from Reserved to In-Use.

[G6] The user must be aware of the cost of the trip during the race.

R25: As soon as the state of the car become In-Use, the counter starts.

R26: During the race, the system should notify to the user the cost of the race.

R27: During the ending procedure, the system stops the counter as soon as the user exits the car, closes the door and all the other doors are closed.

[G7] A user can always end the race inside the service area.

R28: A user can start the ending procedure only if the car is located inside the service area.

R29: Every time the engine is turned off inside the service area, the user must be able to start the ending procedure.

R30: The ending procedure can only begin if the engine is turned off.

R31: When the ending procedure is started, the engine can’t be turned on again.

R32: During a race, if the battery level becomes less or equal than 5%, the system automatically stops the engine and starts the ending procedure if it is not already begun.

R33: The race ends as soon as the ending procedure ends.

R34: When the race ends, the total cost of the race is charged to the user and the state of the car is changed.

R35: Between the end of a race and a new use of the car, the doors can be opened only from inside the car.

R36: Between the end of a race and a new reservation of the car, if any door will be opened, the user of the last race will be charged for the unpaid time between the end of the last ride and the opening of the door.

R37: When a race ends, the system changes the state of the car into Maintenance if the level of battery is less than 40% or the issue flag is equal to true

R38: When a race ends, if the level of battery grows up to 40% and the issue flag is equal to false, the system changes the car state into Available.

[G8] A race with at least 3 people in the car should get a discount of 10% of the total price.

R39: During the ending procedure the system must ask to the user the number of passenger.

R40: If the number of passenger inserted by the user is greater than 1, the system must provide a 10% discount on the race.

[G9] Based on the battery level and the parking position of the car, some extra fees or some discounts can be applied to the cost of the race.

R41: If the race ends and the charger is plugged, the system must provide a 30% discount on the race.

R42: If the car is not plugged into a power grid station and the level of the battery is greater than 50%, user gets a discount of 20% of the cost.

R43: If the car is not plugged into a power grid station the level of the battery of the car is less than 20% user gets an extra fee of 30% of the cost.

R44: If the car is left outside a safe area, user gets an extra fee of 30% of the cost of the race.

[G10] A user can get a discount if he enables the money saving option.

R45: The system must ask the user if he wishes to active the MSO or not before he turns on the engine for the first time.

R46: If the MSO has been activated, the system should allow the user to select his destination through the touchscreen interface.

R47: If the MSO has been activated, the system indicates to the user the best station to park selected by the MSO algorithm.

R48: The system applies the highest discount that user gets to the cost of the race.

R49: The system applies to the cost of the race at most one extra fee.

[G11] A user should be able to report an issue that will be solved by technicians.

R50: The system allows the user to report an issue about the car he reserved or he is using.

R51: If the user reports an issue when the car is reserved, he can choose to start the race or not and if he chooses not to start the race, the system changes the car state to maintenance and the user reservation expires without any costs.

R52: During the race, a user is able to report an issue and continue the ride.

R53: A technician must have access to the cars on maintenance and must be able to solve the issues and set again the issue flag to false.

3.3 Non Functional Requirements

3.3.1 Availability

The system should be available all the time (24/7), granting usage to the users. There must be a dedicated server that will handle the requests. Service maintenance and updates should be informed to the users at least 5 days before.

3.3.2 Multi-Platform

The application should work seamlessly on both the smartphone and the computer. The user should be able to see the reservations and unlock the car accordingly.

3.4 Performance Requirements

At the beginning of the service the system should be able to handle at least 1000 users registered into the database, we consider that a reasonable number of users that are are simultaneously connected is no greater than 10% of the users. Thus the system should be able to handle at least 100 active connections when the system will be deployed.

Moreover, we expect the service to grow rapidly so the architecture of the system should be scalable in order to improve the performances as the number of users and active connections grows.

3.5 Scenarios

Scenarios are narrative description of typical uses of the system. They are written in informal language and they try to represent the point of view of the user. In this case, we assume Bob as a user.

3.4.1 System Registration

Bob is an environmentalist. He is tired to pollute the city with his car. He has to go to work at the other side of the city but he doesn't like to travel using common public transport. So he found over the internet PowerEnJoy, the perfect solution at his problems.
Bob decides to register to the car sharing service. He downloads the application and tries to register to the service. He has to enter: name, surname, email, address, telephone number, driving license and a credit card. He sends the information to the system using the register button. Then he receives an email containing the password needed to login.

3.4.2 Login

Bob, after the success of the registration, decides to login. He enters the email and the password he received after the registration and he presses the login button. After the login Bob is able to use all the services provided by the application.

3.4.3 Pick a car

Bob, wants to use a car to reach the other side of the city. After the login he can see all the available cars near his position using the map of the city. For every available car he can see all the related information (CarID, distance, battery level). Based on it, he chooses the desired one by clicking on the relative "Reserve" button. A message warns Bob that he has 1 hour to reach the car. After that time the system will charges 1€ on his account and his reservation expires. Using the navigation system on the map, Bob reaches the car. When Bob is closer than 20 meters from the car, he opens it using the button shown on the mobile-app. The car opens, the counter that take count of the current cost of the race starts and he is able to enter the car.

3.4.4 Drive the car

After Bob enters the car, he has to answer to some questions showed on the touchscreen. Then Bob starts the engine and sees on the car screen the charges applied for the use of the service.
Bob drive until he reaches the destination.

3.4.5 Park the car

Reached the destination, Bob searches a free power charger to plugs the car. Then he parks, stops the engine, plugs the car and finish the race. At that time, he sees the total cost of the ride on the mobile-app.

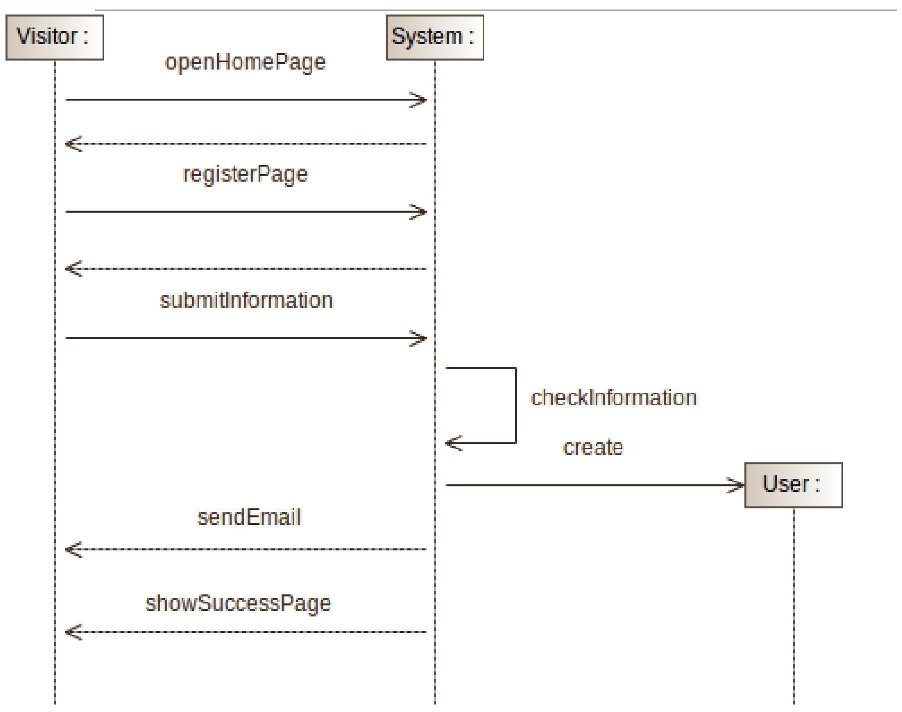
3.4.6 Report an issue

After entering the car, Bob try to start the engine but it doesn't work.
Using the application, he can report the issue. After that, Bob can choose another car using the application.

3.6 Use Cases

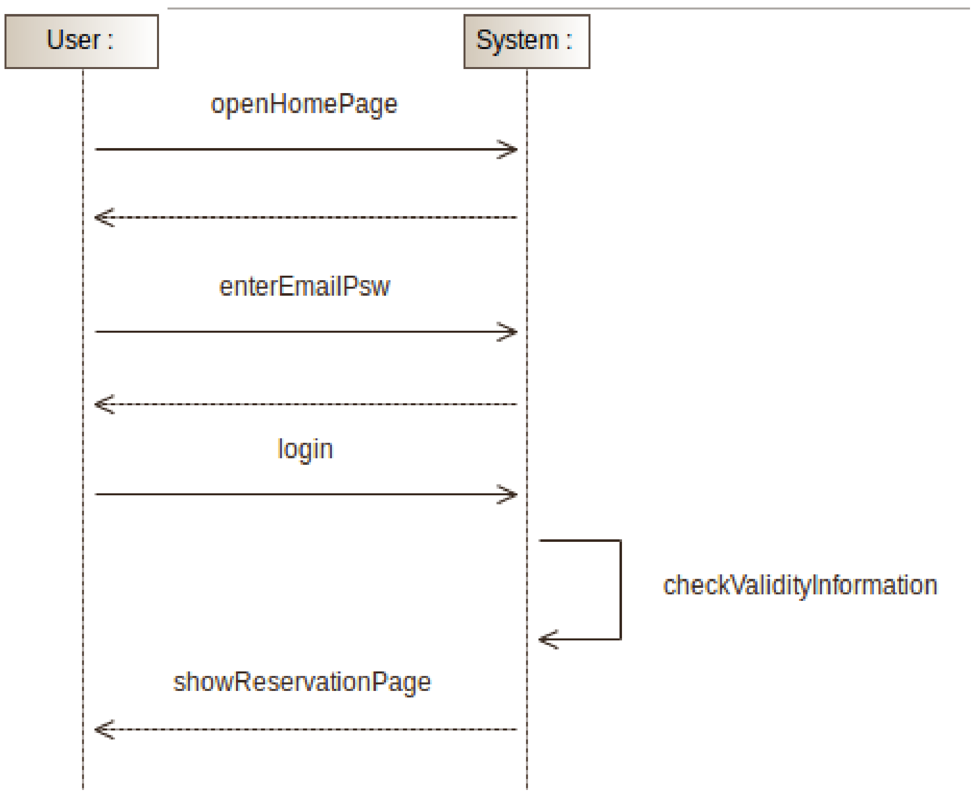
3.6.1 Registering to the system

Actor	Visitor, System
Goal	G1
Input Conditions	Visitor has valid credentials and payment information to submit.
Exceptions	● Visitor didn't fill all the mandatory fields. ● Credentials or payment information are not valid. ● Email is already used by another account. ● The confirmation email is not received.
Flow of Events	1. User connects to the web-app or opens the mobile-app. 2. Clicks or taps the button "Register". 3. The user enters his/her credentials and payment information and submit the form. 4. The system checks the validity of information received. 5. The system creates the new user. 6. The system sends confirmation email containing a random generated password and indicates the success of operation.
Output Conditions	Visitor is successfully registered and becomes a user.



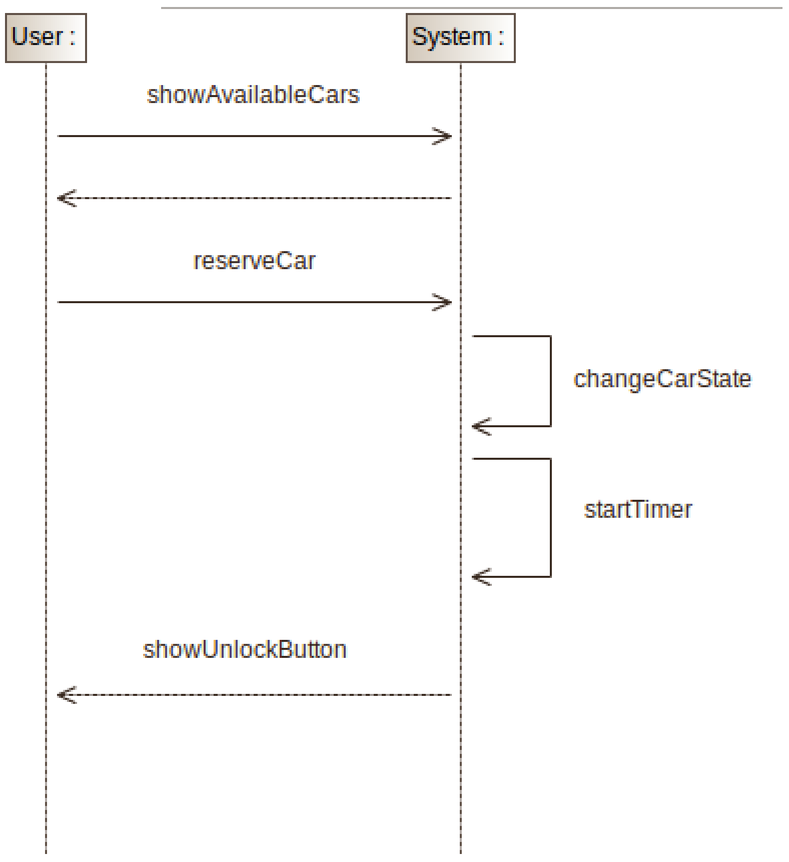
3.6.2 Login

Actor	User, System
Goal	G2
Input Conditions	User must have email and password information.
Exceptions	● The user mistyped password or e-mail. The user will be notified and prompted to enter his credentials again.
Flow of Events	1. User enters the web-app or opens the mobile app. 2. User enters email and password information. 3. User clicks “Login” button. 4. System checks the validity of the information. 5. User is redirected to Ride Reservation page.
Output Conditions	User is allowed to use all the system services.



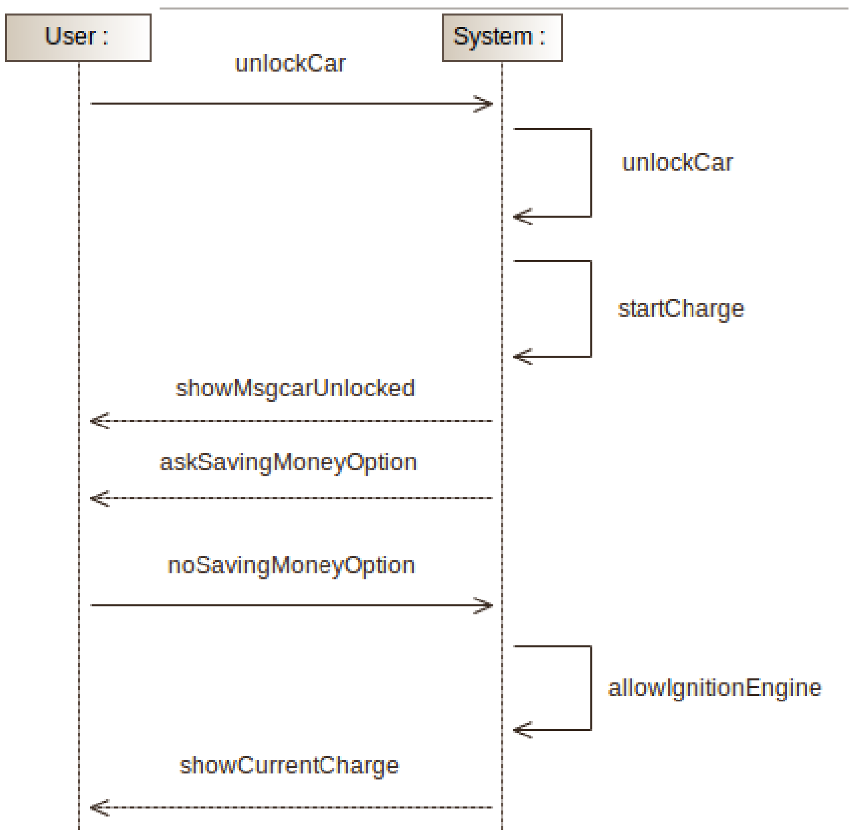
3.6.3 Reserving a Car

Actor	User, System
Goal	G3, G4, G5
Input Conditions	User must be logged in.
Exceptions	● There are not available cars.
Flow of Events	1. User is able to see the available cars and their status near his position or from a given address using the mobile-app or the web-app. 2. User selects a car and taps “Reserve” on the sub menu appearing on selection. 3. The system changes the car state from available to reserved and the Timer starts. 4. The unlock button is now available on the mobile-app.
Output Conditions	The selected car is reserved and waiting to be unlocked by the user.



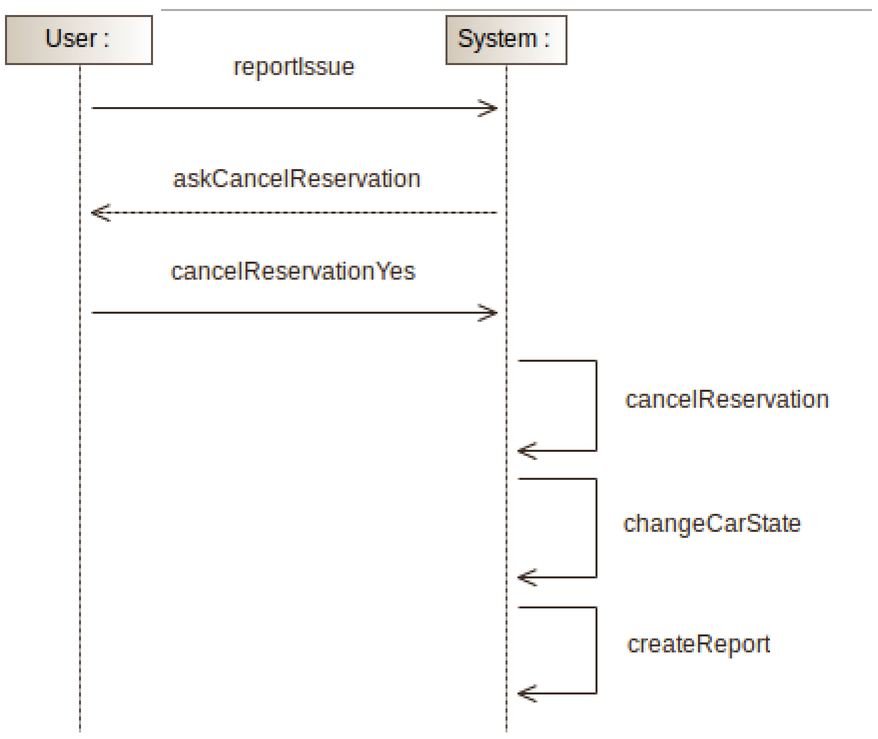
3.6.4 Starting the race

Actor	User, System
Goal	G5, G6
Input Conditions	The car is reserved by the user.
Exceptions	● The user is more than 20m far from the reserved car. The car doesn't respond to the unlock button. ● The reserved car presents an issue, the User report it and choose to cancel the reservation.
Flow of Events	1. User gets in the proximity of the car. 2. User presses the unlock button present on the mobile-app. 3. The system unlocks the car, notify the user and starts charging the user. 4. The user enters to the car and the system offers to activate the saving money option through the touchscreen. 5. The user selects “no” and the system now allows the user to ignite the engine through the power button present on the car. 6. The system shows the current charge of the race through the touchscreen.
Output Conditions	The user can start driving.



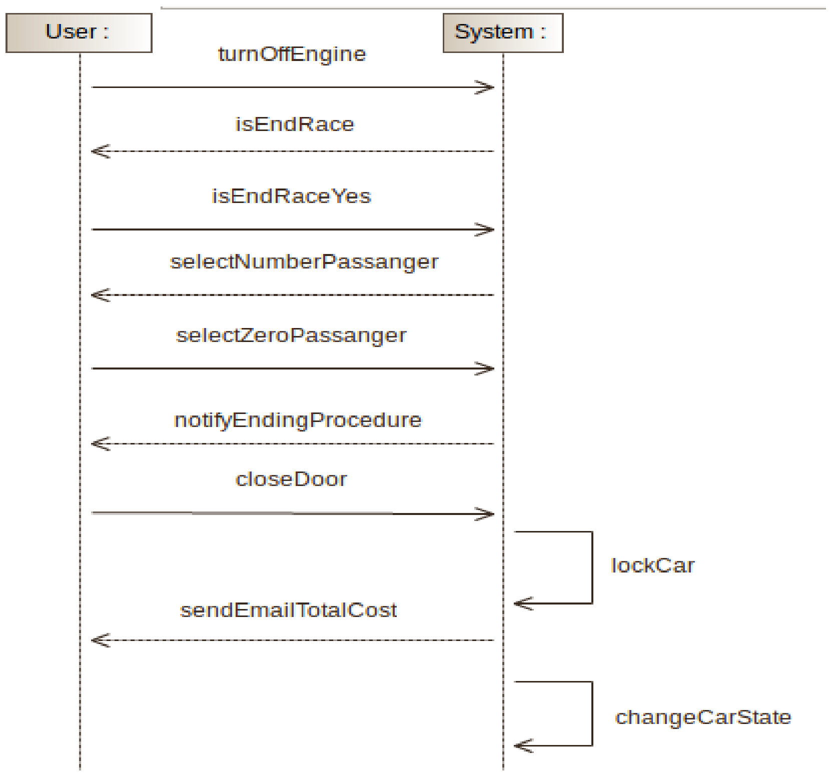
3.6.5 Reporting an issue

Actor	User, System
Goal	G11
Input Conditions	The car state must be “reserved”.
Exceptions	● The car has no problem to be reported. ● The user doesn’t wish to cancel the reservation after reporting the issue.
Flow of Events	1. User reaches the car and notice a major problem on the car. 2. User Taps “Report a problem” from the mobile app and specifies the problem (car located in a private parking spot, broken window etc.). 3. The system asks to the user if he wish to cancel the reservation. 4. The user selects the option “cancel this reservation” 5. The system cancels the reservation of the user for the car and changes the state of the car. 6. The system notifies the technician with a report of the issue.
Output Conditions	● The car state changes to Maintenance. ● The report of the issue is created and available for the technician.



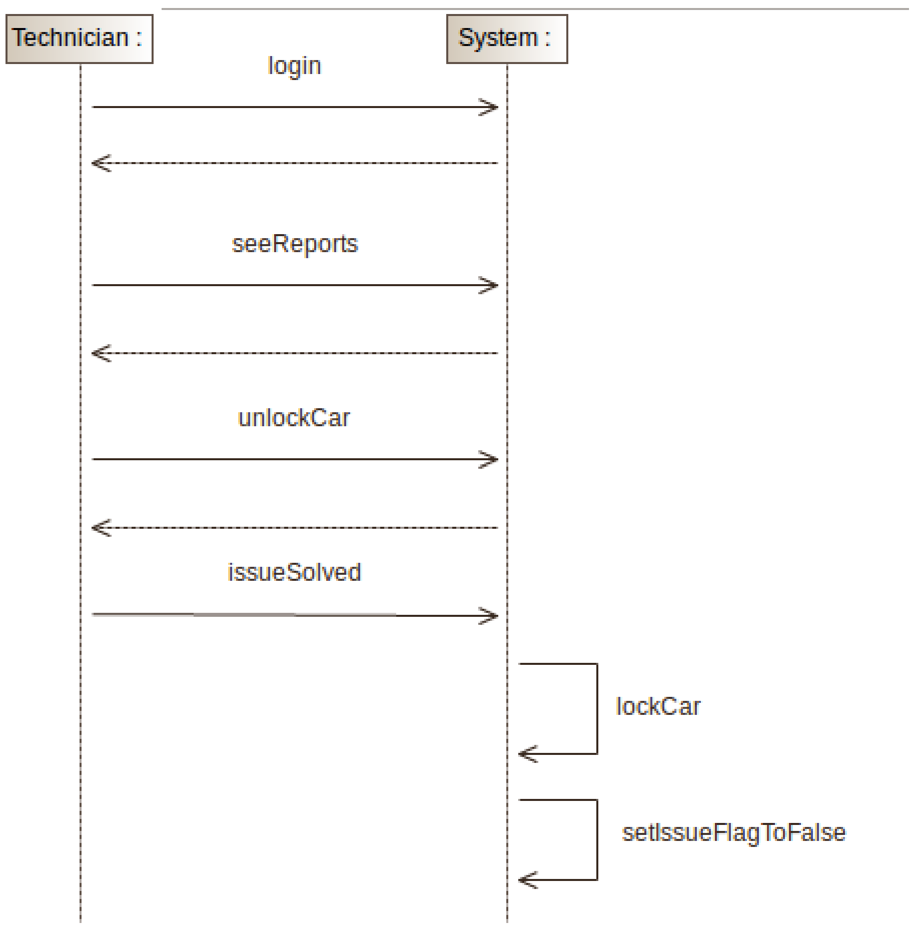
3.6.6 Ending a Race

Actor	User, System
Goal	G7
Input Conditions	The car state is in use.
Exceptions	
Flow of Events	1. The user parks the car inside the service area. 2. The user turns off the engine using the power button. 3. The system asks to the user through the touchscreen if he wants to end the race. 4. The user presses the button corresponding to the end of the race. 5. The system asks to the user through the touchscreen the number of passengers that took part of the race. 6. The user selects zero passengers. 7. The system indicates that the doors will be locked automatically after the driver's door will open and close and all the others doors will be closed. 8. The user exits the car and close the door. 9. The system locks the car and sends an email to the user with the total cost of the ride. 10. The system changes the state of the car.
Output Conditions	● The car's state is not "in use" anymore.



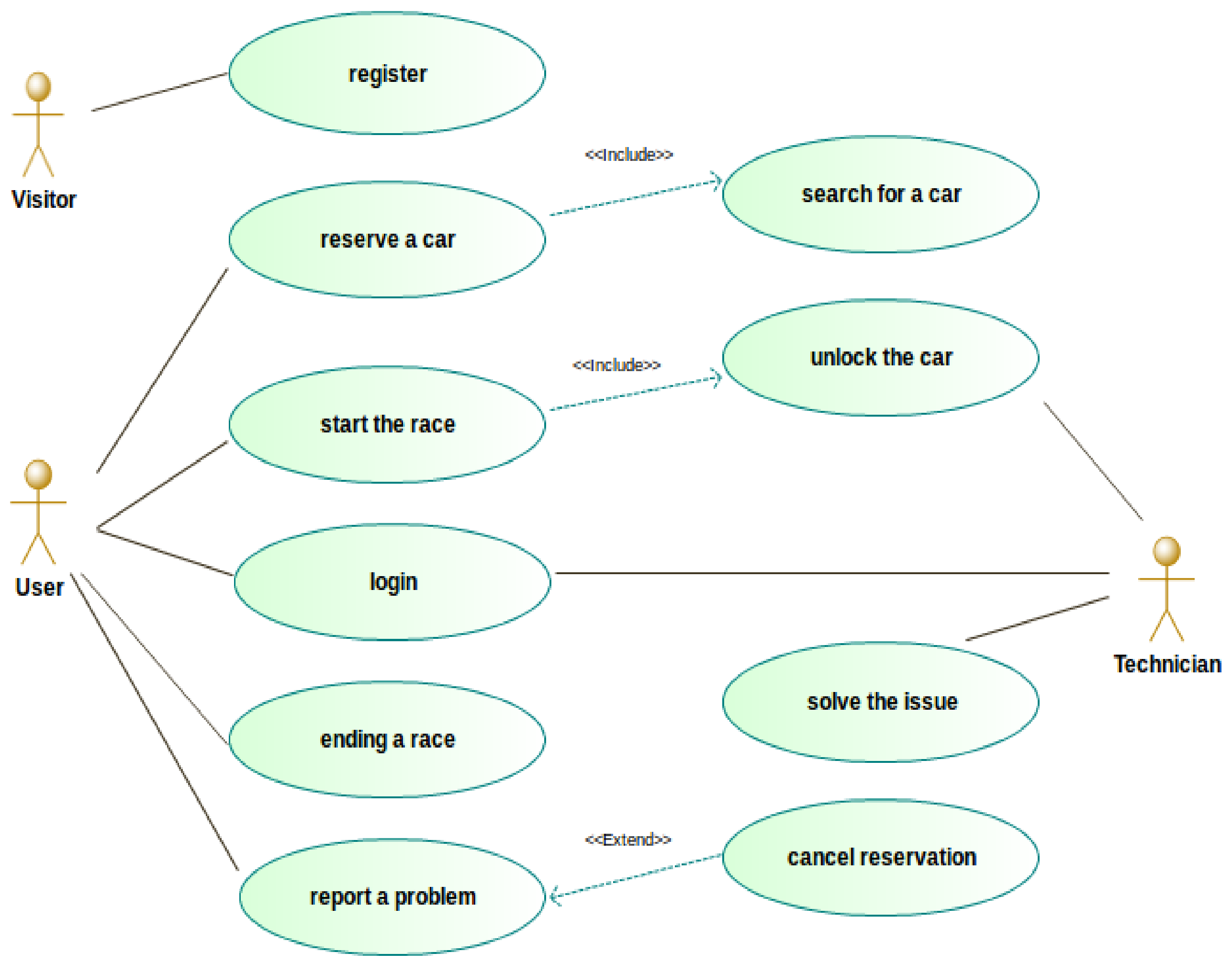
3.6.7 Solving the reported issue

Actor	Technician, System
Goal	G12
Input Conditions	The car state is Maintenance.
Exceptions	● The car's battery is 0%: The car cannot be unlocked. ● When pressing the button "issue solved", some doors are opened: car cannot be locked
Flow of Events	1. A technician logs into the system and see all the reports. 2. Technician chooses the issue he wants to solve and move near the car location. 3. Through the mobile-app he is able to unlock the car and enters inside to solve the problem. 4. Technician exits the car and press the button "issue solved". 5. System immediately locks the car.
Output Conditions	● Issue flag is set to false for the selected car.



3.7 UML Models

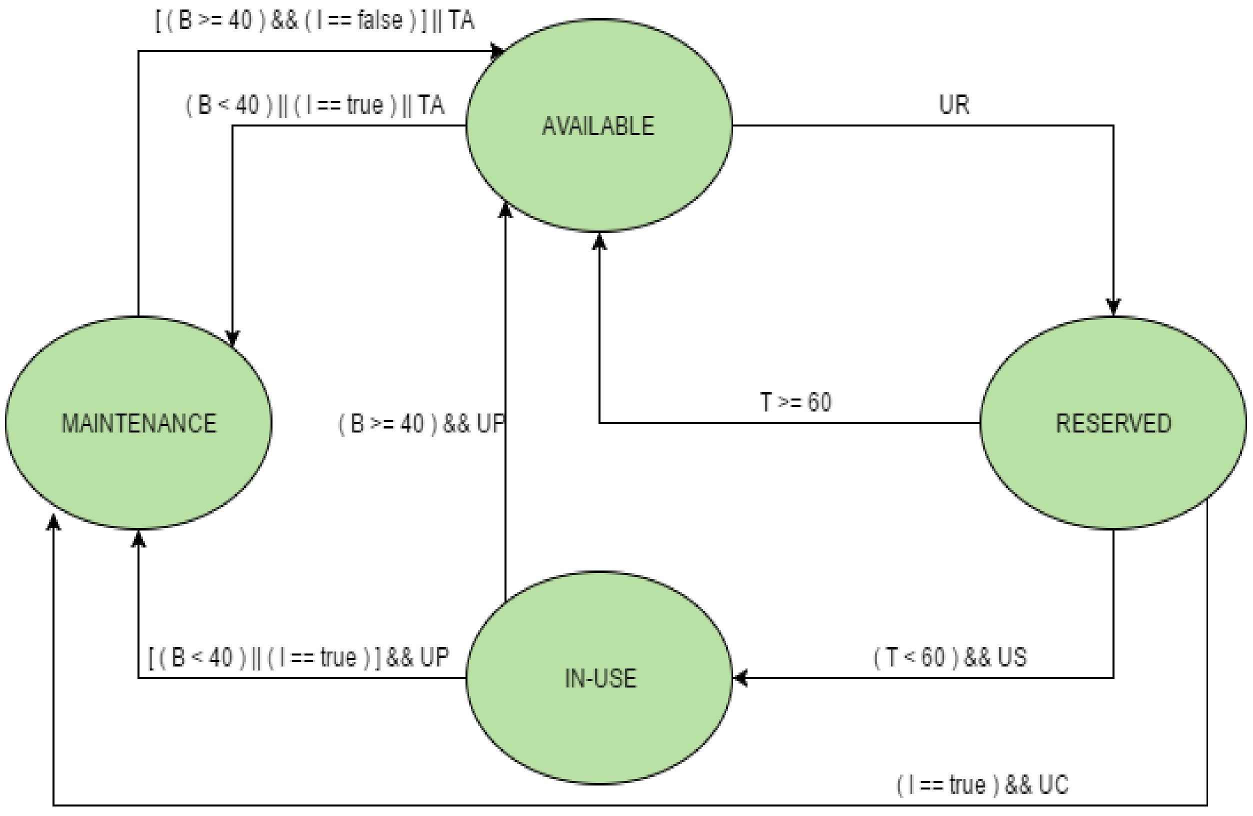
3.7.1 Use Case Model



3.7.2 States of a car

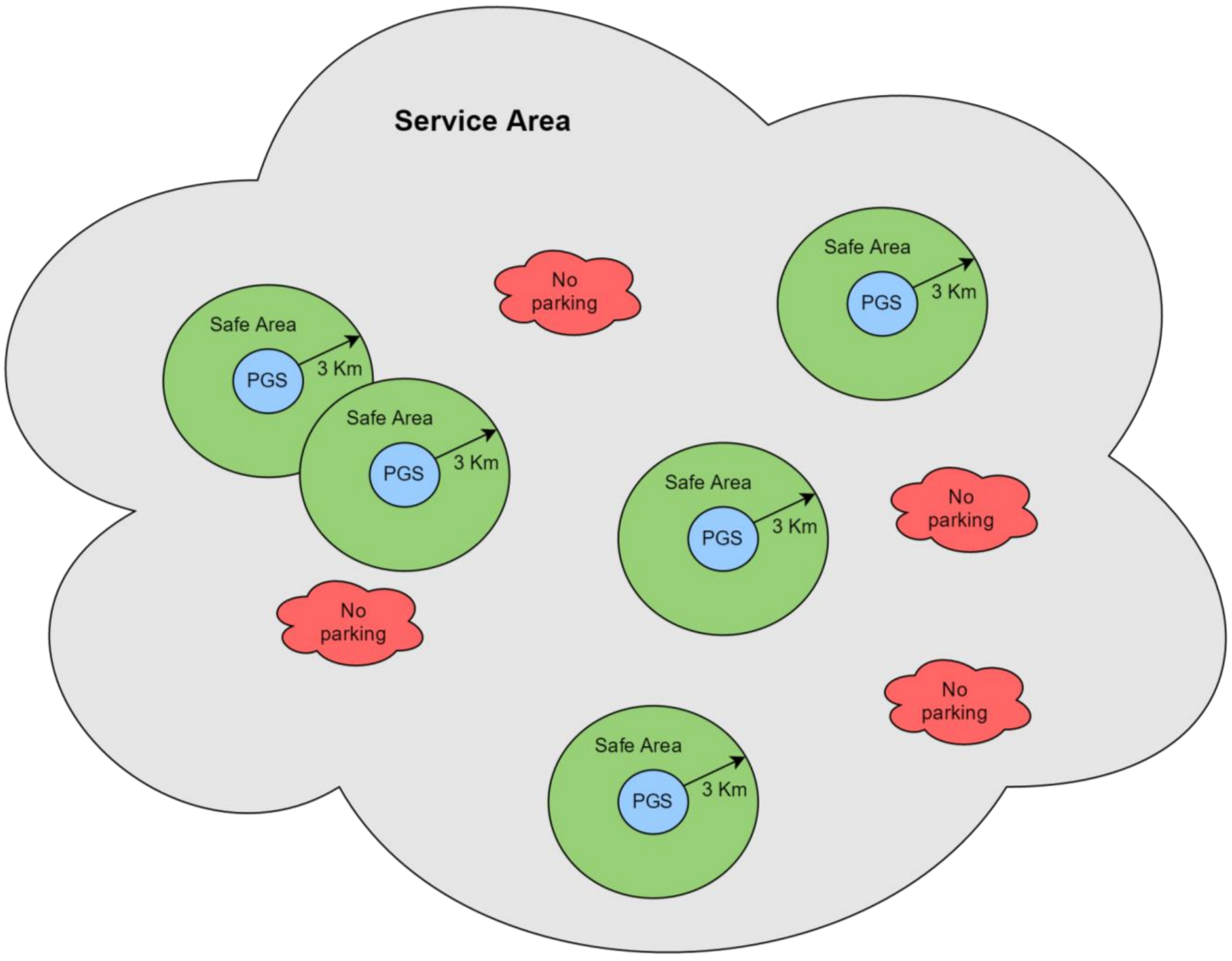
The following diagram helps to understand all the possible states that a car can have and all the transition that could occur.

B = Percentage of the remaining charge of the battery.
I = Issue flag.
T = Timer that count the minutes from the beginning of Reserved state.
UR = User reserve the car.
US = User starts the race.
UC = User cancels the reservation.
UP = User parks the car.
TA = Action by a technician.


[(B >= 40) && (I == false)] || TA
(B < 40) || (I == true) || TA
AVAILABLE
UR
MAINTENANCE
(B >= 40) && UP
T >= 60
RESERVED
[(B < 40) || (I == true)] && UP
IN-USE
(T < 60) && US
(I == true) && UC

3.7.4 Service Map

The following diagram shows how the areas of the system are organized. It is a model that makes an example of the real service map. The real service map is based on the real topology of the city where the service is provided.
The “service area” is the area where a user can park. It is represented by the grey cloud and do not contains the red “no parking areas”. The PGS contains several power chargers, that are not represented in this model.



Chapter 4
Alloy Modelling

4.1 Alloy Code

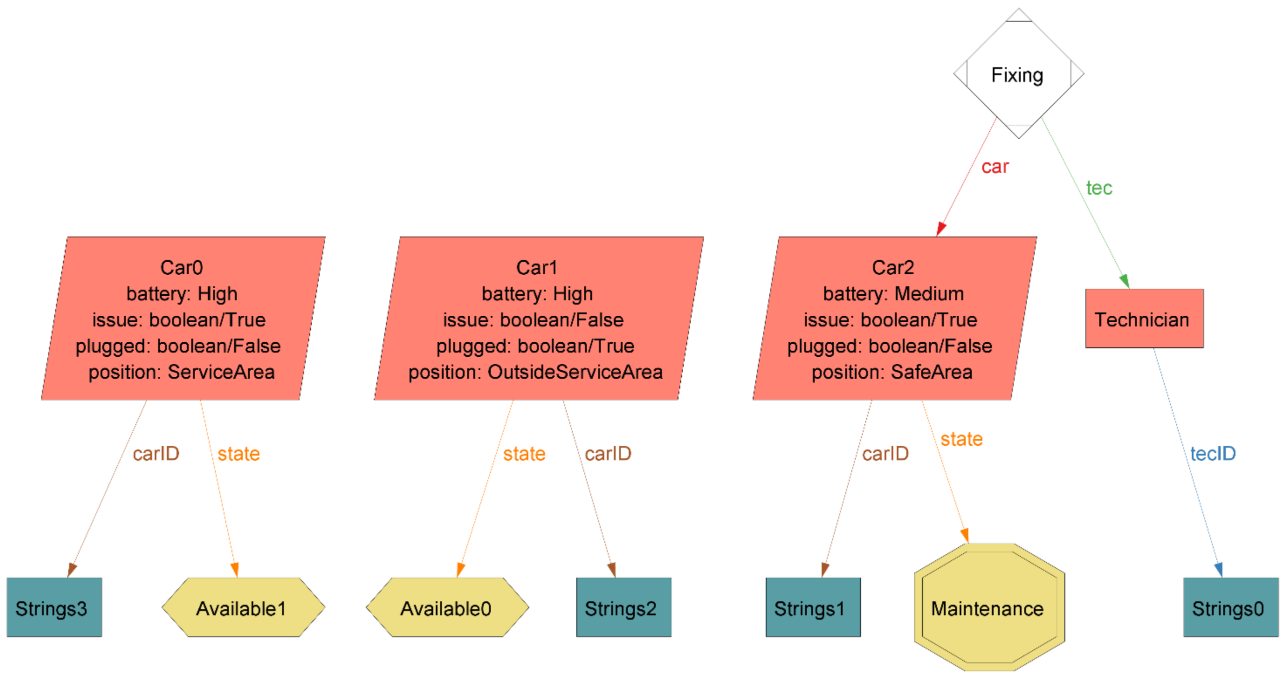
```
---------------------------------- Signatures -----------------------------------

  open util/boolean

  sig Strings{}
  sig Visitor{}

  sig Credentials{
     --  name: one Strings,
     -- surname: one Strings,
     -- licence: one DrivingLicence,
     email: one Strings
  }

  sig PaymentInfo{
     -- owner: one String,
     -- number: one Int,
     -- securityCode: one Int,
     -- expirationDate: one Int
  }
  sig User{
     logged: one Bool,
     paymentInfo: one PaymentInfo,
     credentials: one Credentials,
  }

  /*
     To have an higher level of abstraction we use high, medium and low
     level of battery instead of use the real number used in the system.

     Battery: 0 - 5      = Exhausted
     Battery: 6 - 19     = VeryLow
     Battery: 20 - 39    = Low
     Battery: 40 - 49    = Medium
     Battery: 50 - 100   = High
  */

  enum Battery {Exhausted, VeryLow, Low, Medium, High}
  enum Position {ServiceArea, SafeArea, OutsideServiceArea}
```

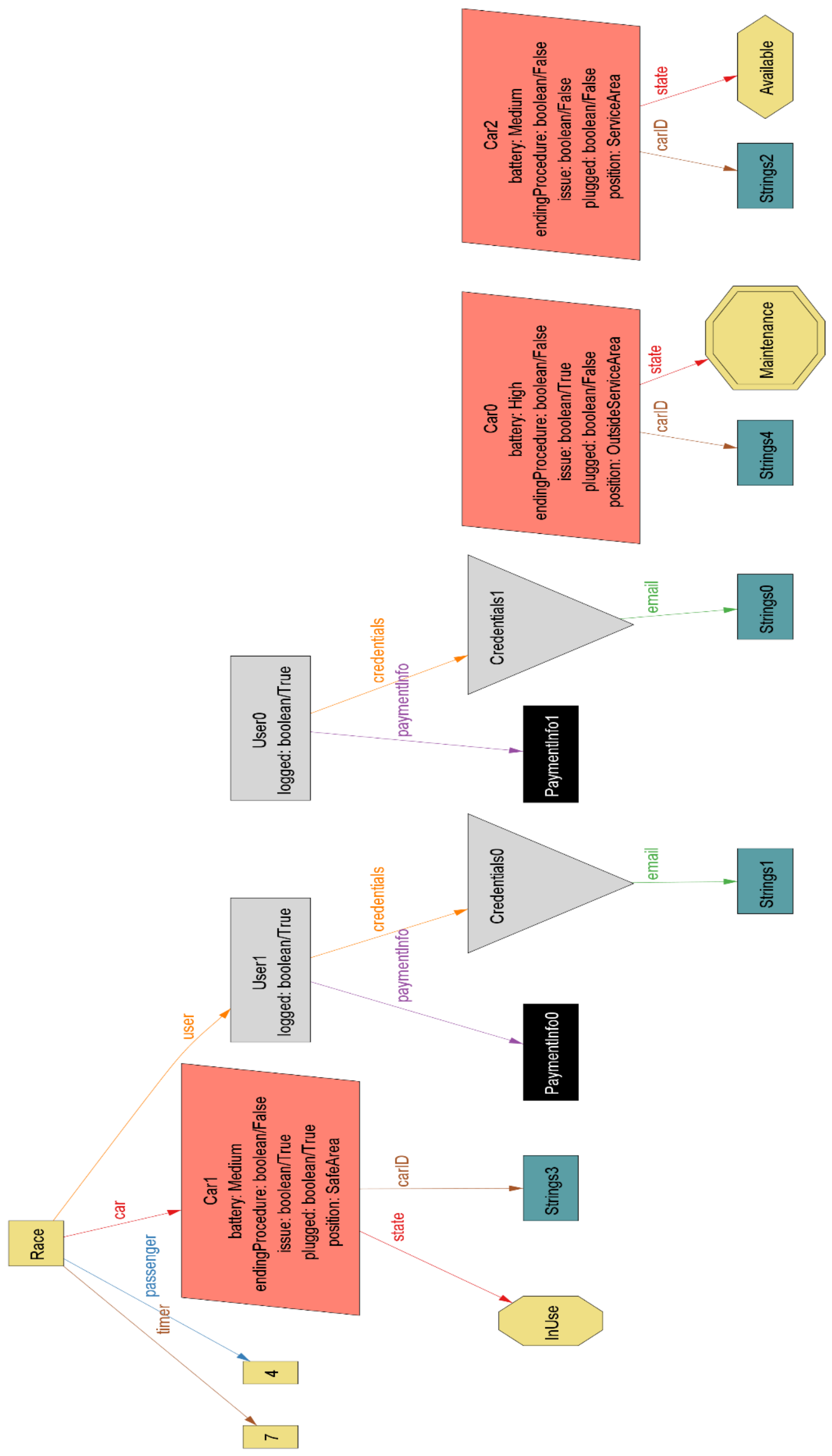
```
abstract sig State {}
sig Available extends State{}
sig Reserved extends State{}
sig InUse extends State{}
sig Maintenance extends State{}

sig Car{
   carID: one Strings,
   battery: one Battery,
   issue: one Bool,
   state: one State,
   position: one Position,
   plugged: one Bool,
}{
   state = Reserved <=> one r:Reservation | r.car = this
   state = InUse <=> one r:Race | r.car = this
}

sig Technician{
   tecID: one Strings
}

sig Fixing{
   car: one Car,
   tec: one Technician
}{
   car.state = Maintenance
   car.issue = True
}

sig Reservation{
   timer: one Int,
   user: one User,
   car: one Car
}{
   timer >= 0
   timer < 60
   car.state = Reserved
}

sig Race{
   timer: one Int,
   user: one User,
   car: one Car,
   passenger: one Int,
   discount: lone Discount,
   extraFee: lone ExtraFee,
   endingProcedure: one Bool
}{
    timer > 0
    passenger >= 0
    passenger < 5
    car.state = InUse
    (endingProcedure = True and (car.battery = Low or car.battery = VeryLow)) => extraFee =
        BatteryLevelExtraFee
    (endingProcedure = True and car.plugged = True) => discount = PluggedDiscount
    (endingProcedure = True and !(car.plugged = True) and car.battery = High ) => discount =
        BatteryLevelDiscount
    one discount => endingProcedure = True
    one extraFee => endingProcedure = True
}

abstract sig Discount{}
abstract sig ExtraFee{}

sig PassengerDiscount extends Discount{}           /*  10%  */
sig PluggedDiscount extends Discount{}             /*  30%  */
sig BatteryLevelDiscount extends Discount   {}     /*  20%  */
sig OutsideSafeAreaExtraFee extends ExtraFee{}  /*  -30%  */
sig BatteryLevelExtraFee extends ExtraFee{}        /*  -30%  */


---------------------------------- Facts ---------------------------------------------------


fact CarConstraints{
    no c:Car | c.state = Available and c.position = OutsideServiceArea
    no c:Car | c.state = Reserved and c.position = OutsideServiceArea
    no disj c:Car | c.state = Available and c.issue = False
    no disj c:Car | c.state = Available and ((c.battery = Medium)||(c.battery = High))
    no disj c:Car | c.state = Maintenance and (c.battery = VeryLow or c.battery = Low) and c.issue = True
    no disj c:Car | c.plugged = True and c.state = InUse and c.position != SafeArea
    no disj c:Car | c.plugged = True and c.state = Reserved and c.position != SafeArea
    no disj r:Race, c:Car | r.endingProcedure = True and c.position = OutsideServiceArea
        and c.battery != Exhausted
    no disj r:Race, c:Car | r.car = c and r.endingProcedure = True and c.state != InUse
}

-- All used Strings{} must be related to a unique other sig
fact NoReusedStrings{
    no disj c:Car, u:Credentials | (c.carID = u.email)
    no disj c:Car, t:Technician | (c.carID = t.tecID)
    no disj t:Technician, u:Credentials | (t.tecID = u.email)
}

-- The carID is unique
fact uniqueCarID{
    no disj c1,c2: Car | c1.carID = c2.carID
}
```

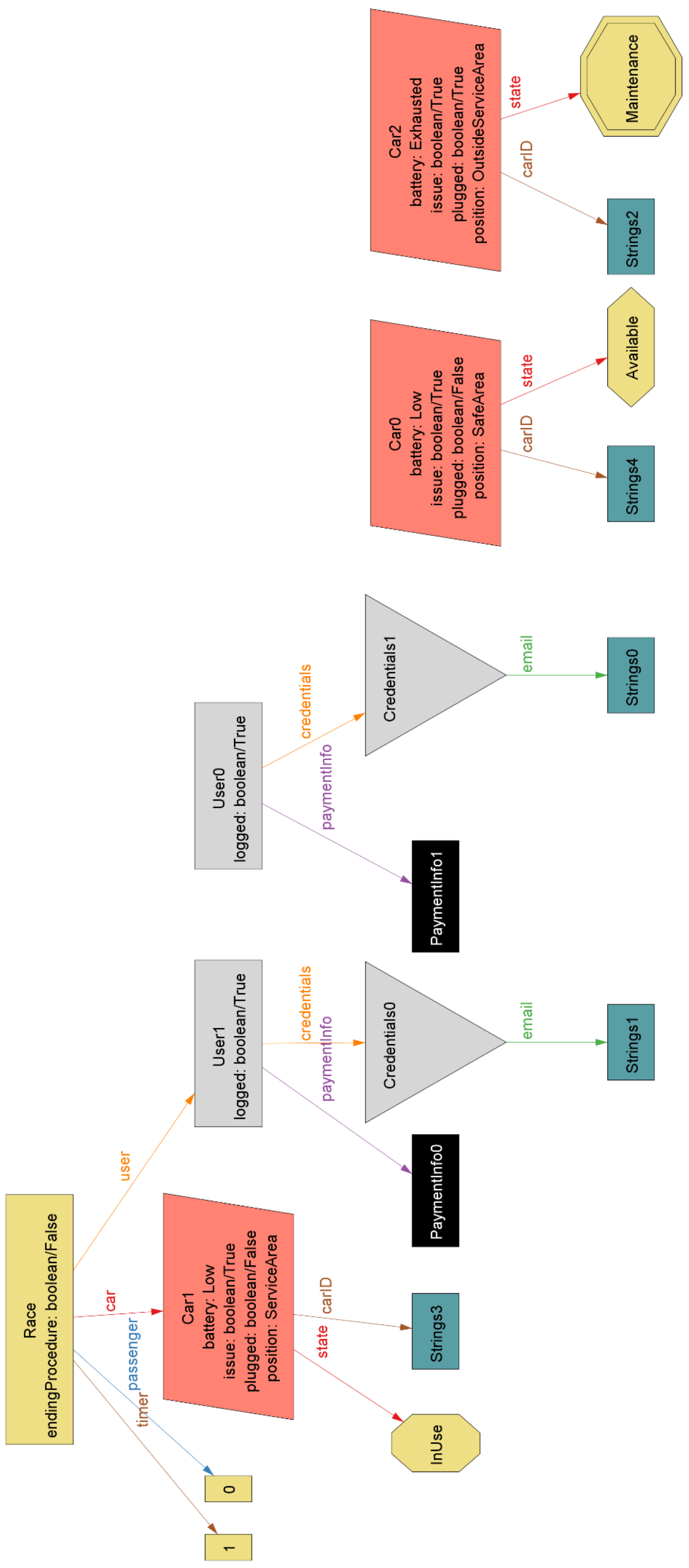
```
-- The Email is unique
fact uniqueUserEmail{
   no disj u1, u2: Credentials | u1.email = u2.email
}

-- The tecID is unique
fact uniqueTecID{
   no disj t1, t2: Technician | t1.tecID = t2.tecID
}

-- One credentials is related to only one user
fact OnePaymentInfoOneUser{
   no disjoint u1,u2: User | u1.credentials = u2.credentials
}

-- One paymentInfo is related to only one user
fact OnePaymentInfoOneUser{
   no disjoint u1,u2: User | u1.paymentInfo = u2.paymentInfo
}

-- Only a logged user can reserve or use a car
fact OnlyLoggedUser{
   no r: Reservation, u:User | r.user = u and u.logged = False
   no r: Race, u:User | r.user = u and u.logged = False
}

-- A car can be reserved only by 1 user at a time
fact OnlyOneReservingUser{
   no u: User | some r1,r2: Reservation | r1!=r2 and u in r1.user and u in r2.user
}

-- A car can be fixed by one technician at a time
fact OnlyOneFixingTec{
   no t:Technician | some f1,f2: Fixing | f1!=f2 and t in f1.tec and t in f2.tec
   no c:Car | some f1,f2: Fixing | f1!=f2 and c in f1.car and c in f2.car
}

-- User can make only one reservation at a time
fact OneReservationPerUserAtATime{
   no u: User | some r1,r2: Reservation | r1!=r2 and u in r1.user and u in r2.user
}

-- User can drive only one car at a time
fact OneRacePerUserAtATime{
   no u: User | some r1,r2: Race | r1!=r2 and u in r1.user and u in r2.user
}

-- Technician can fix only a car at a time
fact OneTecPerCarAtATime{
   no t: Technician | some f1,f2: Fixing | f1!=f2 and t in f1.tec and t in f2.tec
}

-- Discounts and extra fees constraints
fact DiscountsExtraFeesConstraints{
   no d:PluggedDiscount , e:OutsideSafeAreaExtraFee, r:Race | r.discount = d and r.extraFee = e
   no d:BatteryLevelDiscount, e:BatteryLevelExtraFee, r:Race | r.discount = d and r.extraFee = e
   no d:BatteryLevelDiscount, r:Race, c:Car | r.car = c and r.discount = d and c.battery != High
   no d:PassengerDiscount, r:Race | r.discount = d and r.passenger < 2
   no e:BatteryLevelExtraFee, r:Race, c:Car | r.car = c and r.extraFee = e and ((c.battery != Low)
        or      (c.battery != VeryLow))
   no d:PluggedDiscount, r:Race, c:Car | r.car = c and r.discount = d and c.position != SafeArea
        and c.plugged != True
   no e:OutsideSafeAreaExtraFee, r:Race, c:Car | r.car = c and r.extraFee = e and c.position = SafeArea
}

-- No sig without a relation

fact NoStateWithoutCar{
   all s:State | one c:Car | c.state = s
}

fact NoDiscountWithoutRace{
   all d: Discount  | one r: Race | r.discount = d
}

fact NoExtraFeeWithoutRace{
   all ef: ExtraFee  | one r: Race | r.extraFee = ef
}

fact NoCredentialsWithoutUser{
   all c: Credentials | one u: User | u.credentials = c
}

fact NoPaymentInfoWithoutUser{
   all p: PaymentInfo | one u: User | u.paymentInfo = p
}

fact NoPositionWithoutCar{
   all p:Position | one c:Car | c.position = p
}

--------------------------------- Predicates --------------------------------------

pred show(){}

run show for 5
```

4.2 Models Obtained

Here are listed some model obtained with the code above.




Race
car
passenger
timer
user
4
7
Car1
battery: Medium
endingProcedure: boolean/False
issue: boolean/True
plugged: boolean/True
position: SafeArea
carID
state
Strings3
InUse
User1
logged: boolean/True
credentials
paymentInfo
Credentials0
email
Strings1
PaymentInfo0
User0
logged: boolean/True
credentials
paymentInfo
Credentials1
email
Strings0
PaymentInfo1
Car0
battery: High
endingProcedure: boolean/False
issue: boolean/True
plugged: boolean/False
position: OutsideServiceArea
carID
state
Strings4
Maintenance
Car2
battery: Medium
endingProcedure: boolean/False
issue: boolean/False
plugged: boolean/False
position: ServiceArea
carID
state
Strings2
Available


Race
endingProcedure: boolean/False
user
car
passenger
timer
1
0
Car1
battery: Low
issue: boolean/True
plugged: boolean/False
position: ServiceArea
carID
state
Strings3
InUse
User1
logged: boolean/True
credentials
paymentInfo
Credentials0
PaymentInfo0
email
Strings1
User0
logged: boolean/True
credentials
paymentInfo
Credentials1
PaymentInfo1
email
Strings0
Car0
battery: Low
issue: boolean/True
plugged: boolean/False
position: SafeArea
carID
state
Strings4
Available
Car2
battery: Exhausted
issue: boolean/True
plugged: boolean/True
position: OutsideServiceArea
carID
state
Strings2
Maintenance

Chapter 5
Appendixes

5.1 Hours Of Work

Date	Description	Perini	Saini	Türkçapar
18/10/16	Creation of the document structure	1h	-	-
20/10/16	Team work	2h	3h	2h
22/10/16				
25/10/16	Team work	3h	4h	4h
27/10/16				
30/10/16	Doc. structure revision, add contents.	2h	-	-
31/10/16	Use cases	3h	3h	
01/10/16	Use cases		3h	2h
03/11/16	Team work	3h	3h	3h
04/11/16	Chapter 2	3h		
04/11/16	Non-Functional Req., User Interface		2h	4h
05/11/16	2.6 Decisions, diagrams	2h		
07/11/16	Team work	5h	5h	5h
08/11/16	Alloy	2h		
10/11/16	User Interface Fixes			2h
12/11/16	Group Work	5h	5h	5h

12/11/16	Use Cases, User Interface Updates	1h	2h	2h
13/11/16	Release Version	3h	3h	3h

5.2 Document Revisions

This is the first version of the document