Politecnico di Milano
A.A. 2016-2017
Prof. Elisabetta Di Nitto

Students:
Andrea Facchini - 874891
Andrea Milanta - 878403
Antonio Gianola - 877235

PowerEnJoy

Electric Car-sharing Service

REQUIREMENTS ANALYSIS & SPECIFICATION

Andrea Facchini
Antonio Gianola
Andrea Milanta

1. INTRODUCTION 5

1.1 PURPOSE 5
1.2 SCOPE 5
1.3 GLOSSARY 6
1.4 REFERENCE DOCUMENTS 7
1.5 OVERVIEW 7

2. OVERALL DESCRIPTION 8

2.1 PRODUCT PERSPECTIVE 8
2.2 USER CHARACTERISTICS 8
2.3 CONSTRAINTS 9
2.4 ASSUMPTIONS AND DEPENDENCIES 9

3. FUNCTIONAL REQUIREMENTS 10

3.1 PRODUCT FUNCTIONS 10
1. User Identification 10
2. Car Reservation 10
3. Ride Features 11
4. Payment 11
5. Emergencies Management 12
3.2 SCENARIOS 13
Scenario 1 13
William subscribes to PowerEnJoy 13
Scenario 2 13
Julie reserves a car and uses the Money Saving Option 13
Scenario 3 14
Mike forgets about the reservation 14
Scenario 4 14
June has an accident 14
Scenario 5 14
Lucas parks illegally 14
Scenario 6 15
John changes credit card 15
3.3 ACTORS 15
3.4 USE CASES 16
1. Login 17
2. Signing up to the system 18
3. Inserting Payment Information 19
4. Inserting Driving License 20
Sequence diagram 21
4. Reserving a car 22
Sequence diagram 23
5. Cancelling a reservation 24
6. Unlocking the car 25
Sequence diagram 26
5. Starting the car 27
Sequence diagram 28

Andrea Facchini
Antonio Gianola
Andrea Milanta

6. Leaving the car ... 29
Sequence diagram ... 30
7. Setting the MSO mode ... 31
Sequence diagram ... 32
... 32
8. Setting the destination ... 33
9. Charging the AU ... 33
10. Dealing with traffic fines ... 34
11. Emergency alert ... 35
Sequence diagram ... 36
... 36
3.5 CLASS DIAGRAM ... 37

4. NON FUNCTIONAL REQUIREMENTS ... 38

4.1 EXTERNAL INTERFACE REQUIREMENTS ... 38
1. User Interface ... 39
1.1. Mobile and Web Application ... 39
Login Page ... 39
Main Page ... 40
Car screen ... 40
Reservation screen ... 41
1.2. Onboard Computer Application ... 41
PIN screen ... 41
Navigation screen ... 42
Road screen ... 42
2. Hardware Interface ... 43
2.1. Car Interface ... 43
3. Software Interfaces ... 44
3.1. National Transport System Interface ... 44
3.2. Payment System Interface ... 44
PayPal ... 44
Credit Card ... 44
3.3. Onroad Service Interface ... 44
4.2 PERFORMANCE REQUIREMENT ... 45
4.3 SOFTWARE SYSTEM ATTRIBUTES ... 45
Correctness ... 45
Interoperability ... 45
Portability ... 45
Scalability ... 45
Reliability ... 45
Security ... 45
Usability ... 45

5. ALLOY ... 46

5.1 MODEL DESCRIPTION ... 46
5.2 CODE ... 46
5.3 RESULT ... 55
5.4 MODEL GRAPH ... 55
5.5 WORLD INSTANCES ... 56

6. APPENDICES 57

6.1 TABLES 57

1. Cost Table 57
2. Emergency Table 57

Andrea Facchini
Antonio Gianola
Andrea Milanta

1. Introduction

1.1 Purpose

The purpose of the project is to develop a system, named PowerEnJoy Management System, which will make the PowerEnJoy car-sharing service accessible to users.
The system will provide the user with all the available functionalities and will take care of charging the user with the right price, taking into account all the discounts options.

1.2 Scope

The system to be developed has to manage all the functionalities that PowerEnJoy wants to offers to the users. In particular, it will act as an intermediate between the users and the fleet of vehicles.
The main functions of the system will be:

- Managing the users: allow new users to subscribe to the service and already registered users to access the service.
- Allowing Authenticated Users to find a car close to their location and reserve it.
- Managing the reservations: Allow the user to reserve a car up to 1 hour before, and eventually charge him if he fails to reach the car within the hour.
- Charging the user for the ride.
- Managing charge reductions and increments based on the charging table.
- Managing eventual exceptions such as accidents or traffic law fines.

Andrea Facchini
Antonio Gianola
Andrea Milanta

1.3 Glossary

In this document the following acronyms and definitions have been used

- **PowerEnJoy Management System (PEMS):** It's the software system to be developed.
- **User:** The user of the software. He can register as a new user or access to the car sharing service via login and becoming an Authenticated User.
- **Authenticated User (AU):** The User after effectuating a successful login. The AU has access to the service and can reserve a car.
- **Suspended User:** User who has an account but cannot reserve a car because his driving license or payment information are found not be valid.
- **Ride:** (of the vehicle) From when the engine ignites to when the user exits the car at the end of the journey. The length of the ride is measured in minutes and it's calculated by the onboard system.
- **Geographical Region (GA):** Users can only reserve cars parked in this area.
- **Safe parking area (SPA):** Areas where the user is allowed to park.
- **Recharging area (RA):** Areas where the user can put the car in charge.
- **Charging Table (CT):** Set of predefined discount and overcharging options. See the CT in the Appendix, Table 1.
- **Emergency Table (ET):** Set of predefined extra fees to be charged on the user in exceptional situations. See the ET in the Appendix, Table 2.
- **Money Saving Option (MSO):** Option for the AU to ask for a reduction. The system will suggest the user a SPA near the destination and suitable for a discount, according to the CT.
- **Call Center:** Telephone line managed by external
- **Residual Battery (RB):** The battery level of the vehicle.
- **Onroad Service (OS):** People, provided by an external system but at the disposal of the PEMS, which provide autonomous support for exceptional situations directly on site. The OS also provides an emergency call center where the user may directly communicate with an operator.
- **Payment System:** Any generic external system through which a payment to PowerEnJoy happens.
- **National Transport System (NTS):** AU's country state agency that stores all driving licenses information to be validated by external users.
- **Car Statuses:** Terms to synthetically describe the status of the car:
    - **Offline:** not online. When the car is not under supervision of the system. (e.g. during maintenance or Accident).
    - **Free:** Car that is available for reservation, it is not important if it is plugged in or not. This is the standard status for all cars.
    - When the Car that is not available for reservation because it has already been reserved by one user we distinguished this status:
        - **Locked:** Car is locked.
        - **Unlocked:** Car that has been unlocked but ignition is still locked.
        - **Going:** Car that has been started and that the user is driving.

1.4 Reference Documents

- Assignments AA 2016-2017.pdf
- RASD sample from Oct. 20 lecture.pdf
- IEEE standard – WEB.pdf
- Paper on the green move project.pdf
- Second paper on the green move project.pdf
- Oracle Unified Method reference website
  https://blogs.oracle.com/oum/entry/use_case_actors_primary_versus

1.5 Overview

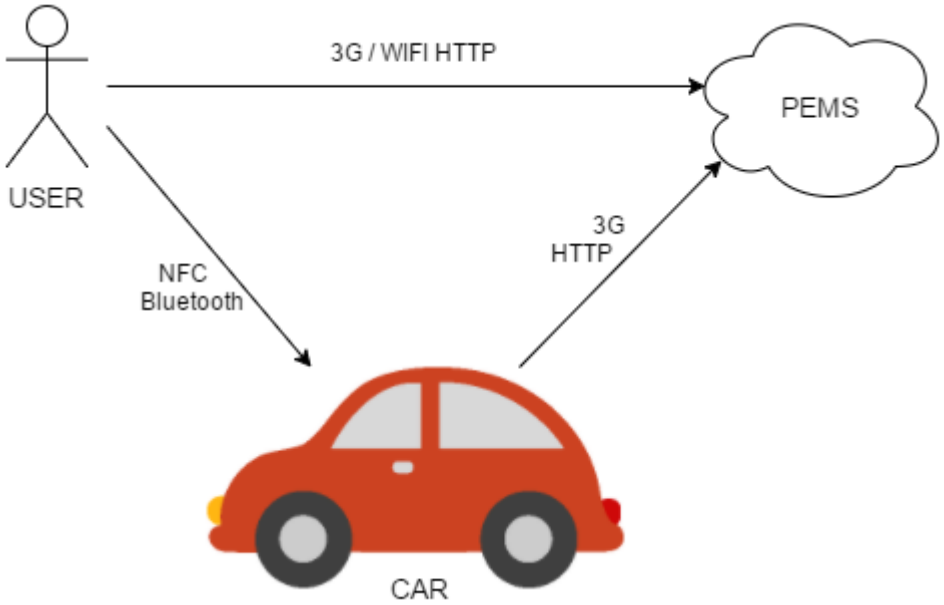
The present document contents the requirements for the PEMS project. This identifies, among others, the domain assumptions, the use cases, the users (stakeholders) and the requirements. The document is organized in the following sections:

1. **Introduction**
   This section introduces the purpose, the scope, clarify the terminology and the acronyms and explain the general aspects of the project.
2. **Overall description**
   This section explains the assumptions, the user's needs, and the requirements, as well as further details on shared phenomena and the domain model.
3. **Functional Requirements**
   This section analyzes more in depth the functional requirements of the project.
   It contains the different functional requirement, use cases, scenarios, and sequence diagrams.
4. **Non Functional Requirement**
   This sections describes all those requirements which do not directly concern the functionalities of the system.
   It contains analysis of the external interfaces and of the software attributes.
5. **Alloy**
   This sections describes the proposed Alloy model and the result of the analyzer.
6. **Appendices**
   This sections contains additional materials which may be useful to the reader.

2. Overall Description

2.1 Product Perspective

The new system is strongly based on a client-server architecture.
The system is completely self-managed, thus there is no need for a system administrator.
Both mobile and web application will be dealt by the same server using the MVC pattern.
To manage the reservations steps and the cars a state machine has been defined.
Although most of the system is software, the project will cover also some hardware development to simplify the interface with the user within the car.
Cars are unlocked directly by the user via NFC or Bluetooth, hence removing delays due to server requests.
There are no constraints on the number of cars and GAs.



2.2 User Characteristics

The main target of the PowerEnJoy service are young adults, from legal driving age (16-18 depending on the country) to the mid-thirties. These category of costumers is usually technology-friendly and mostly oriented towards mobile interfaces.
Also they usually act fast, meaning that they require a service to be ready at hand and fast to use.
While young adults represent the main target, the system allows anyone with a valid driving license to use the service. Subsequently it is expected to have some low percentage of older people using the system.
This calls for a very clear user interface since these customers usually have less dexterity with technology.

2.3 Constraints

Different types of constraints have been identified

- **Regulatory policies imposed by government institutions**
  - A valid driving license is required for any user to drive a car.
  - All “sensible” information, such as the user’s name, surname, driving license ID and payment information, can only be accessible by the authorized users.
- **Hardware limitations**
  - Cars are equipped with an onboard computer whose interface may vary a lot depending on the car model and manufacturer.
  - Car batteries go flat. Once a car has an empty battery it cannot, in physical terms, be used by the service.
- **Safety and security consideration**
  - The system is required to protect its user from any dangerous situation and provide support for emergencies.

2.4 Assumptions and Dependencies

The document describing the problem has not a univocal interpretation regarding some functionalities.
Hence in the development of the project the following assumptions have been made.

1. Each GA is a set of SPA.
2. There are no SPA which do not belong to any GA.
3. Each RA is within a SPA.
4. The user is allowed to pick up a car in a certain GA and leave it in another GA.
5. The user is allowed to drive anywhere.
6. In case of accident or empty battery out of a safe area, the PEMS will provide support at the user’s expenses.
7. Fines will be charged to the user.
8. Cars with more than 90% of battery empty will be Offline.
9. Any NTS is always right when sending traffic fines.
10. If more than a discount is applicable, only the biggest one (as in more reduction) is applied.

All requirement can only be considered valid if these assumptions are true.

3. Functional Requirements

3.1 Product Functions

The product functions can be divided into 5 main areas:

1. User Identification

All those functions required to properly and uniquely identify a user

	Requirement	Related Use Case
1.a	Users must be uniquely identifiable by the system.	Login Signing up to the system
1.b	The user must insert a valid driving license to prove he is allowed to drive a car. (see Regulatory Policies constraint i)	Inserting driving license
1.c	A car can only be unlocked and driven by the reserving user.	Unlocking the car Starting the car

2. Car Reservation

All those functions required to properly and uniquely identify a user

	Requirement	Related Use Case
2.a	The user must be able to check the distance of any available car from his position or from a specified address.	Reserving a car
2.b	The user must be able to reserve a car	Reserving a car
2.c	Once reserved a car must be available for the user to use for up to one hour.	Reserving a car
2.d	The user must be able to categorize the available cars according to the distance from his position or a specified address.	Reserving a car
2.e	The user must be able to cancel a reservation before the one-hour time limit	Cancelling a reservation

Andrea Facchini
Antonio Gianola
Andrea Milanta

3. Ride Features

All those functions that allow a user to properly reserve, unlock, drive and lock a car.

	Requirement	Related Use Case
3.a	The system must be able to provide the riding user with the MSO.	Setting the destination Setting MSO Mode
3.b	The system must be able to provide the riding user with navigation to his intended destination.	Setting the destination

4. Payment

All those functions that allow money transactions to be executed between PowerEnJoy and the Payment System of the user.

	Requirement	Related Use Case
4.a	The due fees must be charged on the user automatically.	Inserting payment information Charging the AU
4.b	The user who just ended a reservation must be charged the due fee, taking into account all the discount options.	Leaving the car
4.c	The user who has reserved a car but not reached it within the one-hour limit must be charged with a 1€ fee.	Reserving a car
4.d	Traffic fines must be charged on the user driving the car at the moment of the infringement	Dealing with traffic fines
4.e	In case of emergency, when a user is involved, the system must charge the user the due fee according to the emergency table.	Emergency Alert
4.f	Car reservation is not allowed until all due fees are paid off.	Charging the user Reserving a car

5. Emergencies Management

All those functions that allow the PEMS to deal with emergencies and malfunctions

	Description	Related Use Case
5.a	Accidents, improper parking and empty battery must be detected automatically by the system.	Emergency Alert
5.b	The OS can notify the system of a malfunction or emergency	Emergency Alert
5.c	In case of emergency or malfunction the system must provide support by means of the OS.	Emergency Alert

Andrea Facchini
Antonio Gianola
Andrea Milanta

3.2 Scenarios

Scenario 1

William subscribes to PowerEnJoy

William is a foreign student who decides to register to the PowerEnJoy car-sharing service to be able to use a car for shopping in town. He downloads the PowerEnJoy app from the online store and access the registration page. He enters his personal information (email, Name, Surname, address and birthdate) and chooses a password. He is then granted access to the service and receives his personal code via email. He accesses his profile page and inserts his payment information, his PayPal credentials, and his valid license id and expiration date. Now William has full access to the service and can reserve a car anytime he needs one.

Scenario 2

Julie reserves a car and uses the Money Saving Option

Julie has already registered to the service. She needs a car to go to a doctor appointment on the other side of town. She accesses the PowerEnJoy app and sees all the cars close to her location. She chooses one approximately 500 meters far, and taps the *Reserve Car* button. A one-hour countdown is then displayed on the app, together with the *Unlock Car* button and the navigation to the car. Julie tries to unlock the car from her position, but she is notified that she is too far for that operation. She then starts walking to the car. After a few minutes she gets to the car and tries to unlock it again. This time the car unlocks and she enters it. Once inside the car she enters her personal code on the onboard displays and starts the car. She knows the way to the doctor so she does not need navigation, she decides though to use the Money Saving Option. She taps the *Money Saving Option* button on the onboard display and inserts her destination address. She is suggested a Recharging Area about 400 meters from her actual destination that will give her a 30% discount. She accepts the suggestion and the display starts providing navigation to the chosen area. Julie drives to the recharging area. Once parked in the RA, the display confirms that the car is parked in a valid location and reminds Julie to plug-in the car to get a discount. Julie exits the car, inserts the plug and leaves. After one minute she receives an email with the paid fee and the confirmation of the end of the reservation.

Scenario 3

Mike forgets about the reservation

Mike has reserved a car to go to the movies with his girlfriend and some friends. Just as he's about to leave the house, he receives a text from his girlfriend telling him she's sick. He gets worried, text his friends they won't be going and forgets about the car.
After 45 minutes he receives an email from PowerEnJoy reminding him of his reservation, but he is too worried to read it. After one hour he receives an e-mail stating that the reservation has been ended and he has been charged 1€ for not having cancelled the reservation before.
In the meanwhile, Mike's friends, who had already reserved a car as well, decided not to go without Mike. Hence the one who reserved a car accessed the app and cancelled his reservation by tapping the *Cancel Reservation* button.

Scenario 4

June has an accident

June is a student who has just been clubbing with her friend. It is 4am and she decides to use PowerEnJoy to get home because no public transport is available.
Via the app she reserves a nearby car and quickly gets to it. After having entered the car, she inserts her private code and her home address to get navigation information.
She starts driving, but after a few minutes of driving alone she falls asleep and the car goes into a ditch. Luckily she is not hurt, but she panics. No one is passing by and she does not know what to do. After a few minutes a tow trucks gets to the site. She is both relieved and surprised. The OS Officer informs her that an automatic emergency alert had been emitted by the car and he is there to take care of the situation. The OS Officer calls a taxi for June and takes the car to a suitable workshop. A little later June receives an email from PowerEnJoy where she learns that a 250€ fine has been charged on her account due to the accident.

Scenario 5

Lucas parks illegally

Lucas is driving a PowerEnJoy car. He has reserved it to go to the stadium and is now approaching his destination. The traffic is very heavy and it is not easy to find a parking spot. He then decides to leave it on the hard shoulder of a dual carriage road which passes close to the stadium. Lucas parks the car with the hazards on and leaves the car without noticing the onboard display showing that he is not parked in a Safe Parking Area.
After a few minutes Lucas receives an email notifying him that the car has been left out of a SPA and if it is not moved within 15 minutes he will be charged a 250€ fine. Lucas hence decides to go back to the car and look for a valid spot. After a few minutes he finds a good place and parks the car. The reservation is then ended normally and after one minute he receives an email with the paid fee and the confirmation of the end of the reservation.

Scenario 6

John changes credit card

John has just ended a reservation like many times before. Instead of receiving the normal end of reservation email though, he receives an email where he is informed that the payment of the due fee was unsuccessful and he will not be able to use the service again until it is paid successfully. At first John is surprised, but rapidly realizes that he just changed his credit card and has not updated such information on the app. Being at home, he browses to the PowerEnJoy website, logs in with his credential, and goes to the Payment Information page. He updates his credit card data and clicks the *Pay Due Fee* button. Little after he receives a new email notifying him of successful payment and that he is now able to use the service again.

3.3 Actors

According to the Oracle Unified Model definition for “Actor” the following actors have been identified

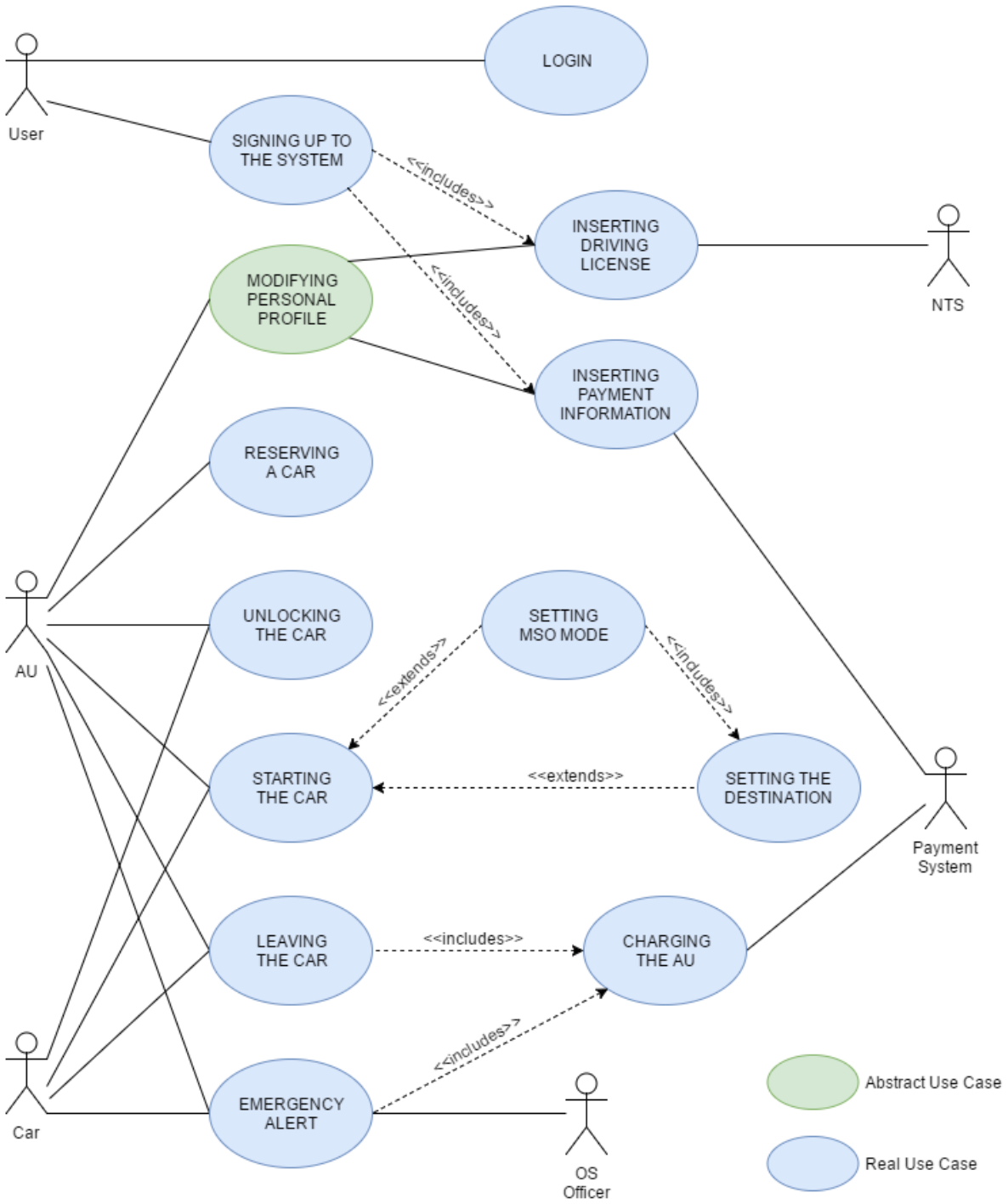
- User
- Authenticated User
- Car
- Payment System
- Onroad Service
- National Transport System

Actor: An actor specifies a role played by a user or any other system that interacts with the subject. [Oracle Unified Method]

3.4 Use Cases

From the study of the scenarios the following use cases have been identified

In many use cases there is more than one participating actor, hence it has deemed simpler to show just one Use Case UML Model for the whole system.



Andrea Facchini
Antonio Gianola
Andrea Milanta

1. Login

Login	
Overview	This use case describes how the user access the service
Actors	User
Entry Conditions	No user is already logged in the device the user is using to login
Flow Of Events	The user gets to the login page The user inserts his ID and password and the presses the *login* button The System grants the user access to the service
Alternate Flow Of Events	None
Exit Conditions	The User becomes an AU
Exceptions	If the credentials are not valid or the user is already logged in another device the process is interrupted and the user is notified. The user is then allowed to reinsert the credentials.

Andrea Facchini
Antonio Gianola
Andrea Milanta

2. Signing up to the system

Signing up to the system	
Overview	This use case describes how the user signs up to the PowerEnJoy service
Actors	User
Entry Conditions	The user must have downloaded the mobile app or browsed to the website
Flow Of Events	The user gets to the registration page The user inserts his personal information. These are; • Full Name and Surname • email address • BirthDate • Country of Residence • Age The user chooses a password for his account. The user inserts his payment information (see 3. Inserting Payment Information Use Case) The user inserts his driving license (see Inserting Driving License Use Case) The System creates his account and sends the user his private code required to start cars.
Alternate Flow Of Events	None
Exit Conditions	The user is registered but suspended
Exceptions	If the email is already associated to an existing account, the registration is invalidated and the user notified. The user is then allowed to try again.

3. Inserting Payment Information

Inserting Payment information	
Overview	This use case describes how the user inserts credentials for the automatic payment of the rides.
Actors	Authenticated User, Payment System
Entry Conditions	The User must be logged in or Signing up
Flow Of Events	The user accesses the payment information page. The user chooses his payment information between credit card or PayPal. The user inserts his payment information: - Credit card number, owner, expire date and CSV if he chooses credit card. - Mail and Password if he chooses PayPal. The System check correctness. The system stores all information in a secure way.
Alternate Flow Of Events	If the driving license is also valid the user is allowed to use the service and is no longer suspended.
Exit Conditions	The payment information is registered on the system.
Exceptions	If the payment information is not valid the user is allowed to reinsert the credentials.

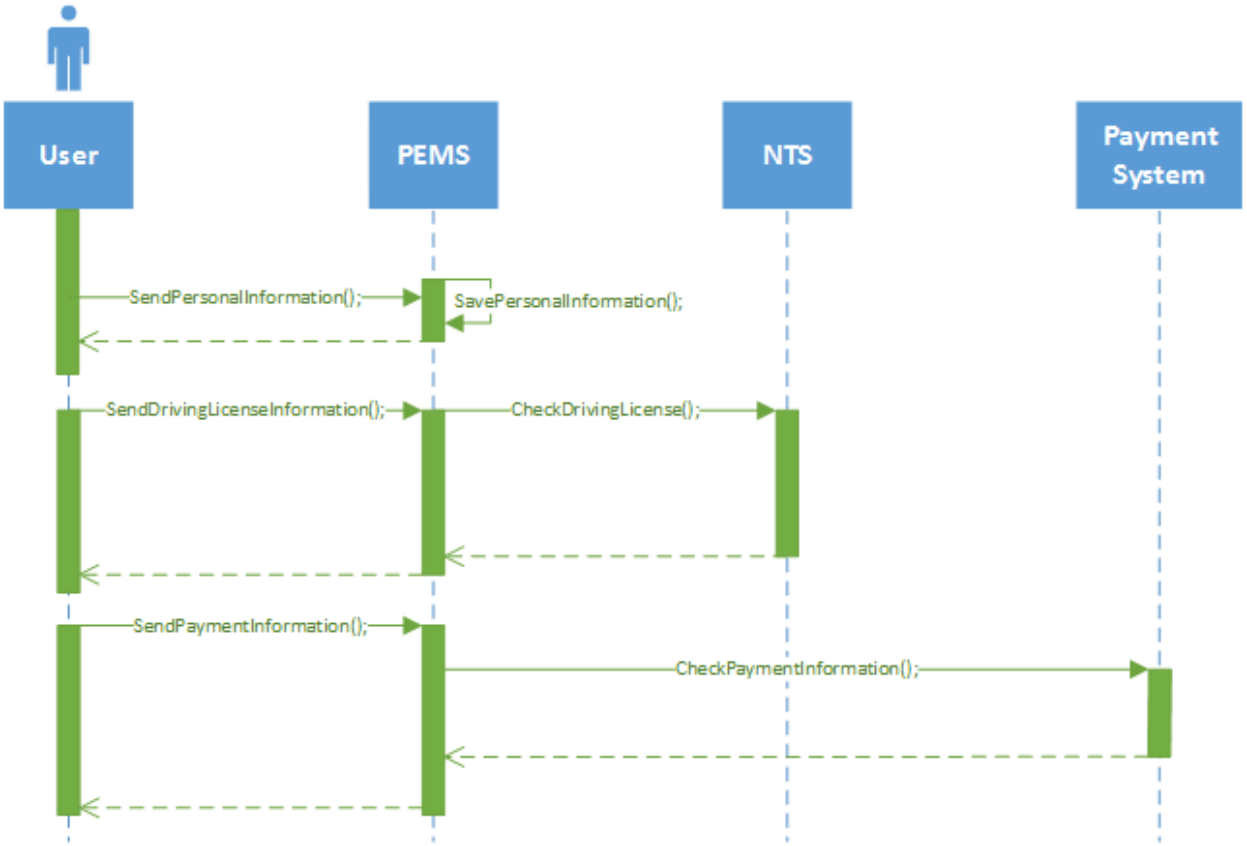
4. Inserting Driving License

Inserting Driving License	
Overview	This use case describes how the driving license of the user can be inserted and validated by the system
Actors	Authenticated User, National Transport System
Entry Conditions	The User must be logged in or Signing up
Flow Of Events	The user accesses the modify driving license page The user inserts his driving license identification code, expiration date and country of emission. The system checks for the information to be valid on the NTS of the user's country of residence. The system checks that the user name and surname match those of the driving license. The system stores all information in a secure way.
Alternate Flow Of Events	If the payment information is also valid the user is allowed to use the service and is no longer suspended.
Exit Conditions	The driving license is registered on the system.
Exceptions	If the driving license is not valid or does not match the user's credentials the user is notified and allowed to reinsert the credentials.

Sequence diagram

This sequence diagram follows the use cases:

1. Signing up to the system
2. Inserting payment information
3. Inserting Driving License



Andrea Facchini
Antonio Gianola
Andrea Milanta

4. Reserving a car

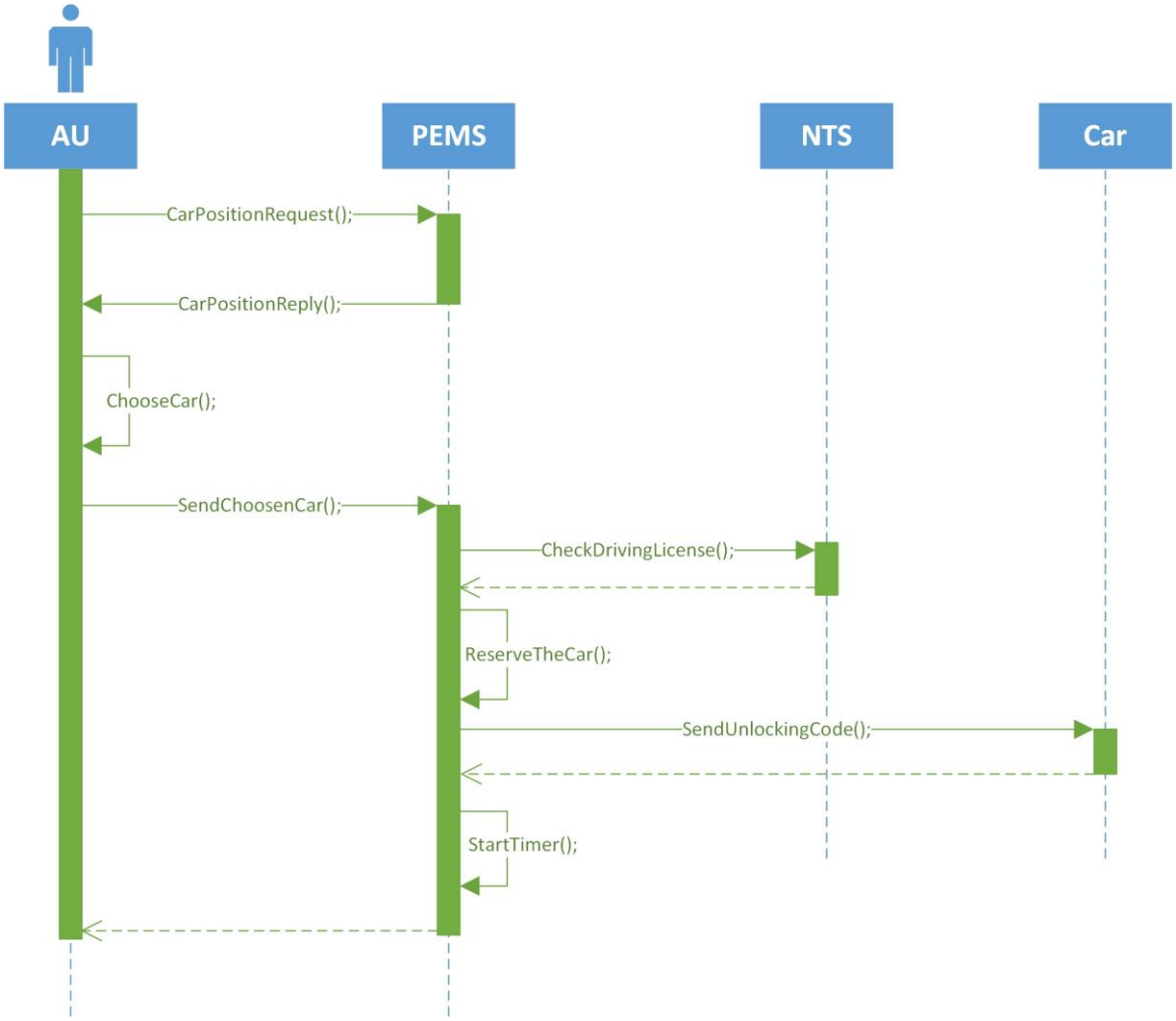
Reserving a car	
Overview	This use case describes how the AU reserves a car
Actors	AU
Entry Conditions	The user must not be using a car and must have not already reserved one
Flow Of Events	The user can see all the available cars in his default GA. The user chooses a car and can see all the relevant information including distance from his position. The user then confirms the reservation. The system removes the car from the available ones and starts a timer.
Alternate Flow Of Events	The user chooses to see only cars within a certain distance from his position or a specific address. The cars are then restricted to those which meet the criteria. If it is the user's first reservation of the day, the driving license is checked to be valid. (see Use Case License validation). The PEMS sends to the car the reserving AUs reference. If the car is not accessed within 45 minutes the system notifies via email the user that he only has 15 minutes left to get to the car.
Exit Conditions	The car is not available for other users to reserve and a timer is started.
Exceptions	If the driving license is found not to be valid the process is cancelled and the user is notified. The user is then suspended until he provides a valid license. If the user is suspended the user can browse the cars but he is not allowed to reserve one. If the user does not unlock the car within one hour the reservation is ended and the user is charged with a 1€ fee. (see Unlocking the car and Charging the AU Use Case)

Sequence diagram

This sequence diagram follows the use cases:

4. Reserving a car

We send unlocking code to the car in order to unlock the car with NFC without send a message to the server.



Andrea Facchini
Antonio Gianola
Andrea Milanta

5. Cancelling a reservation

Canceling a Reservation	
Overview	This use case describes how the user cancels a reservation
Actors	User
Entry Conditions	The AU has already reserved a car but not yet entered it.
Flow Of Events	The user presses the *Cancel Reservation* button. The system registers the requests and cancels the reservation The AU is then allowed to reserve a new car
Alternate Flow Of Events	None
Exit Conditions	The reservation is ended
Exceptions	None

Andrea Facchini
Antonio Gianola
Andrea Milanta

6. Unlocking the car

Unlocking the car	
Overview	This use case describes how the AU unlocks the car and gains access to the vehicle.
Actors	AU, Car
Entry Conditions	The car is locked and has been reserved by the user less than one hour before
Flow Of Events	When the user gets to the car he places his device on the *Unlock* NFC plate mounted on the car. The car unlocks the door and notifies the system. Once the user accesses the car (door / seat sensors) the system is notified.
Alternate Flow Of Events	When the user gets to the car he presses the button *Unlock Car.* The system notifies the car to unlock the door. Once the user accesses the car (door / seat sensors) the system is notified.
Exit Conditions	The user is in the vehicle.
Exceptions	If the user presses the unlocking button when he is too far away the request is rejected and the user notified. If there is any kind of failure in the unlocking process (mechanic, communication, etc..) the user is notified. If the user doesn't get into the car within one minute from unlock the car locks. In all the exception the user is then allowed to try again.

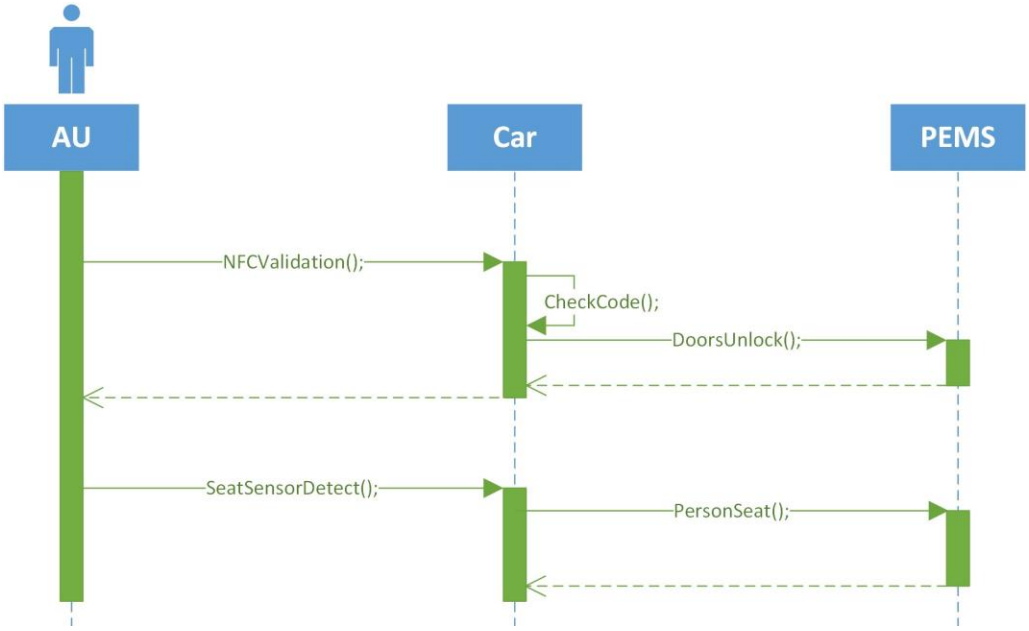
Andrea Facchini
Antonio Gianola
Andrea Milanta

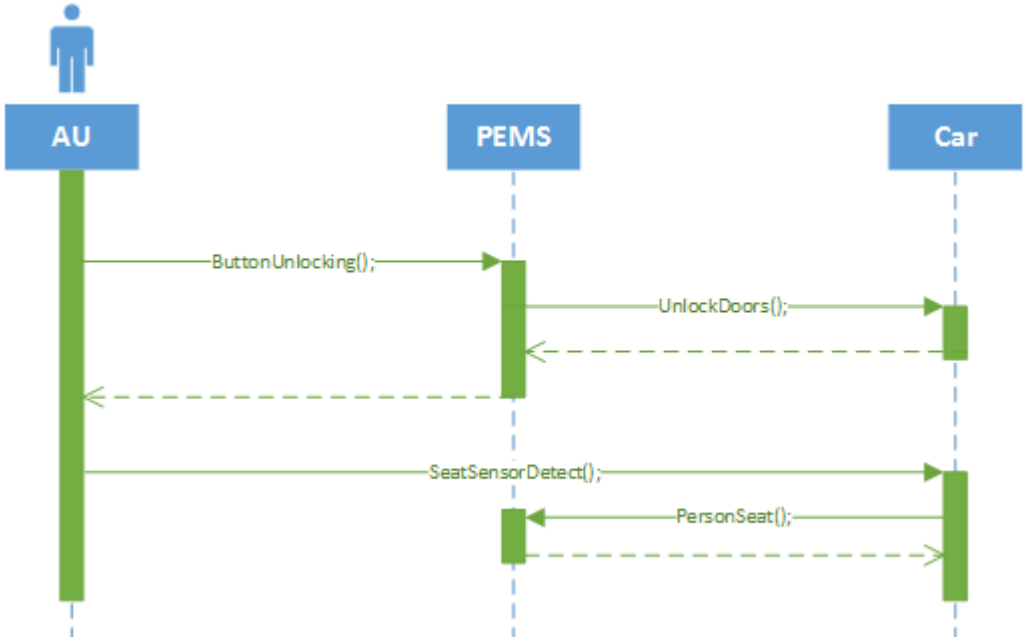
Sequence diagram

This sequence diagram follows the use cases:

4. Unlocking the car

The first diagram shows the unlock with NFC.
The second diagram shows the unlock with *Unlock Car* Button.





Andrea Facchini
Antonio Gianola
Andrea Milanta

5. Starting the car

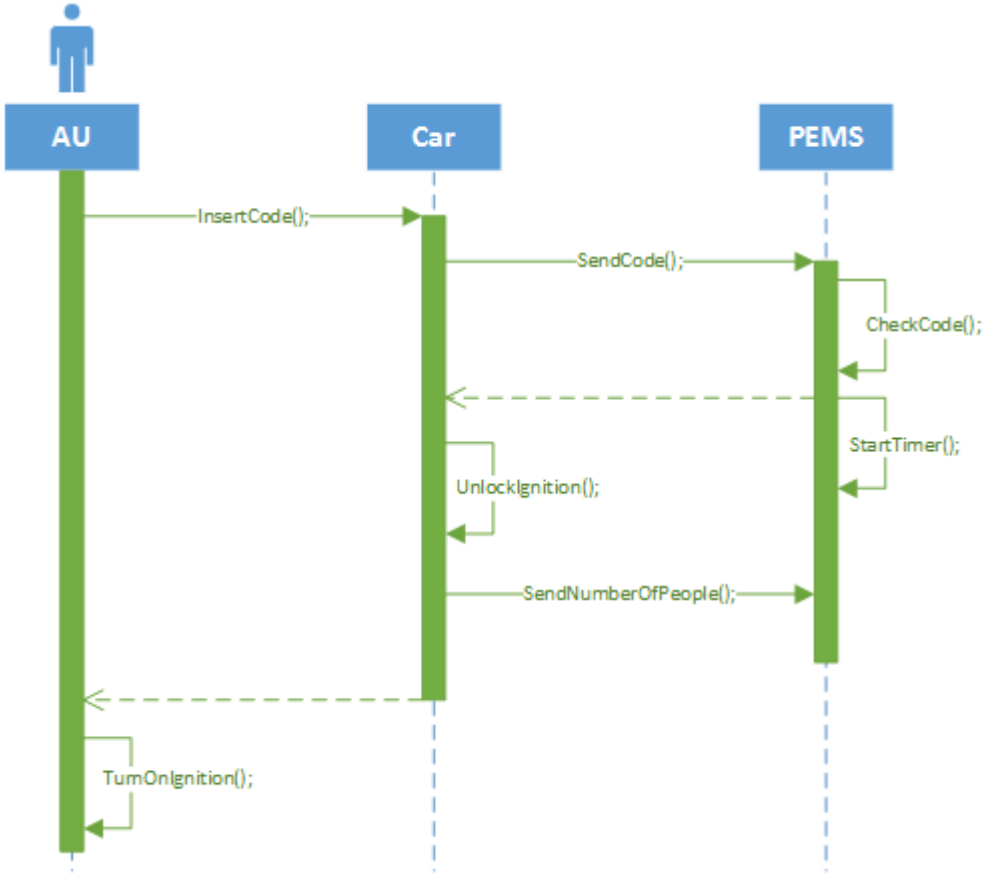
Starting the car	
Overview	This use case describes how the user starts the car
Actors	Authenticated User, Car
Entry Conditions	The user just got into the car, Ignition is off and locked
Flow Of Events	The user inserts its personal code on the onboard display. If no code is inserted, after one minute a beep and a visual warning reminds the user to insert the code. The car sends the code to the system for verification. Once the car receives notification from the system of a valid code, it unlocks the ignition. The system records the time. The system also records the number of people in the car (seat sensors). The onboard display shows the current basic charge. The user can then turn on the ignition and start driving.
Alternate Flow Of Events	None
Exit Conditions	The user is in control of the car.
Exceptions	If the wrong code is inserted the ignition is kept locked and the user is allowed 2 more attempts. If all attempts are unsuccessful or no code is inserted within two minutes from entering the car an Emergency Alert is automatically submitted to the system. (see Use Case Emergency Alert)

Andrea Facchini
Antonio Gianola
Andrea Milanta

Sequence diagram

This sequence diagram follows the use cases:

5. Starting the car



Andrea Facchini
Antonio Gianola
Andrea Milanta

6. Leaving the car

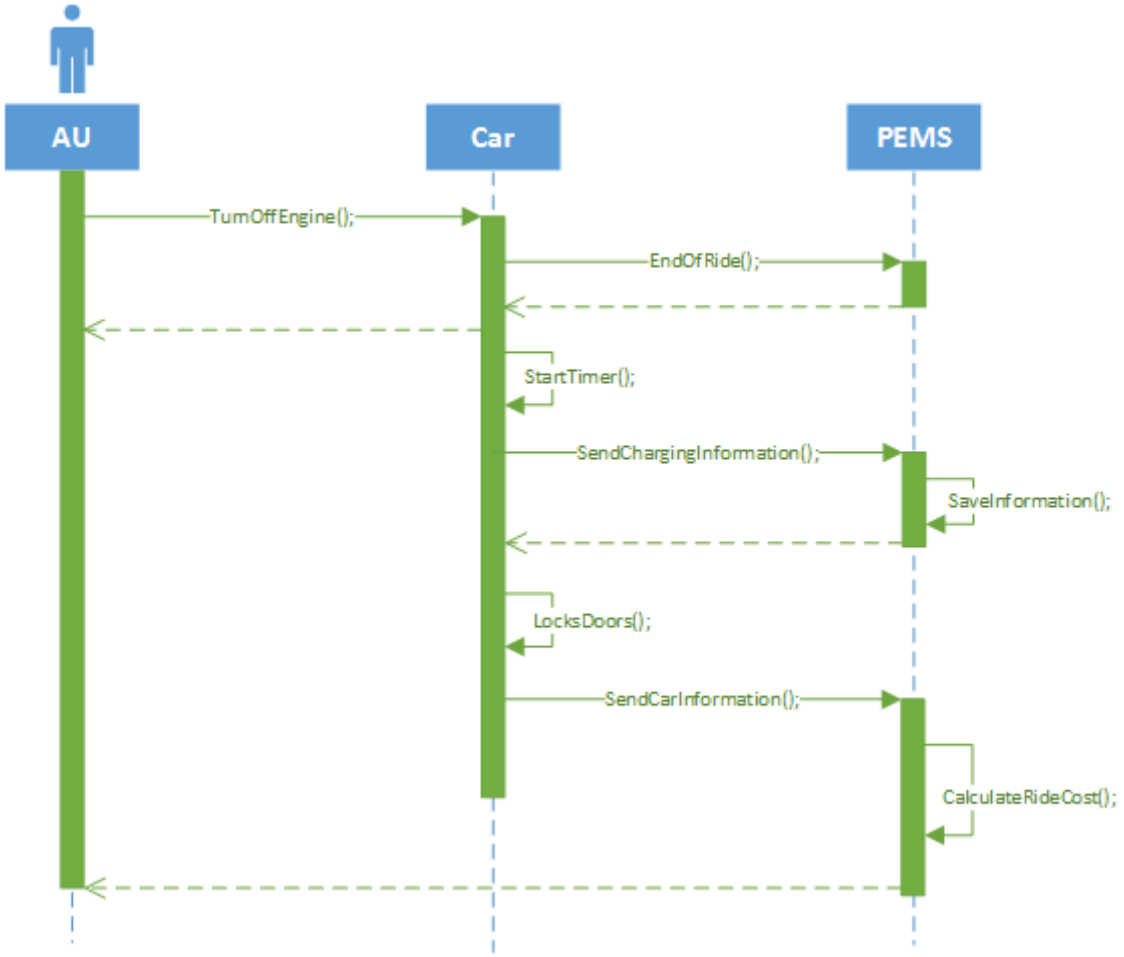
Leaving the car	
Overview	This use case describes how the AU leaves the car and the reservation is ended
Actors	Authenticated User, Car
Entry Conditions	The car must already have been reserved and unlocked, and it is stopped with the engine turned off.
Flow Of Events	The System notifies the car that it is parked in an SPA and a relevant message appears on the onboard display. The user exits the car. The car waits a minute. The car then locks the door and the ignition The car notifies the system of the new condition, and the final battery status. The PEMS calculates the ride time and the following basic fee. The PEMS calculates the due discount according to the charging table, based on the discounts criteria met by the ride. The registration is ended and the car is returned to available. The user is notified of the end of registration and the final fee.
Alternate Flow Of Events	If the car is parked in a RA and the user has attached the charger, the system is notified of this as well. If the battery is more than 90% empty and the car is not recharging the system alerts the OS, the registration is ended but the car goes in the exceptional status . If the battery is more than 90% empty but it is charging, the system will end the reservation but only return the status to Available after the battery goes above 10%.
Exit Conditions	The reservation is ended and the car returned to available.
Exceptions	If the car is not in an SPA the car remains in the active condition and the user is notified via both onboard display and sms. If the door or ignition locking fails the car remains in the active condition and the user is notified via both onboard display and sms. If the exceptional condition is not resolved by the user within 15 minutes an emergency alert is emitted by the car (see Use Case Emergency Alert)

Andrea Facchini
Antonio Gianola
Andrea Milanta

Sequence diagram

This sequence diagram follows the use cases:

6. Leaving the car



Andrea Facchini
Antonio Gianola
Andrea Milanta

7. Setting the MSO mode

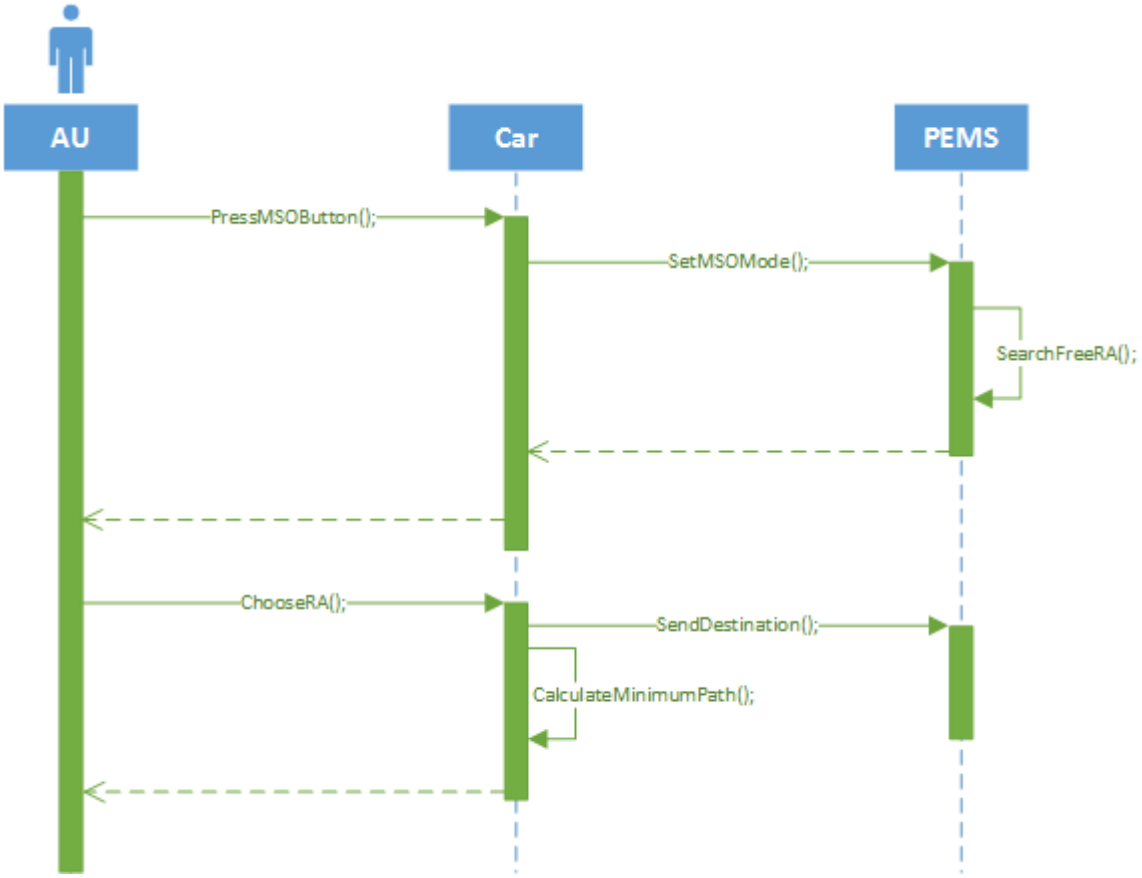
Setting MSO Mode	
Overview	This use case describes how the AU to use MSO mode.
Actors	Authenticated User
Entry Conditions	The AU has inserted a valid code.
Flow Of Events	The AU presses the MSO button on the car display. The AU inserts his destination. (see use case). The PEMS searches for RAs close to the set destination where the user can leave the car to get a discount, and prints them on the onboard display.
Alternate Flow Of Events	The user can choose a proposed RA. The car display provides navigation to the chosen RA.
Exit Conditions	None.
Exceptions	None.

Andrea Facchini
Antonio Gianola
Andrea Milanta

Sequence diagram

This sequence diagram follows the use cases:

7. Setting MSO Mode



Andrea Facchini
Antonio Gianola
Andrea Milanta

8. Setting the destination

Setting the destination	
Overview	This use case describes how the AU sets the destination of his ride.
Actors	Authenticated , Car
Entry Conditions	The AU has inserted a valid code.
Flow Of Events	The AU chooses the destination by typing an address or by searching directly on the map. The car calculates and shows the shortest path to the destination.
Alternate Flow Of Events	if the AU type an address out of GA the car display notifies the user.
Exit Conditions	The shortest path to the destination is showed on the car display.
Exceptions	None

9. Charging the AU

Charging the AU	
Overview	This use case describes the payment of the due fee
Actors	Payment System
Entry Conditions	The AU has an unpaid fee.
Flow Of Events	The PEMS charge the AU via his preferred Payment System.
Alternate Flow Of Events	None
Exit Conditions	The AU is charged the due fee.
Exceptions	If the payment is unsuccessful the AU is notified via e-mail and gets suspended until the payment is successful. The AU can choose to pay unsuccessful transactions at any moment by pressing the *Pay Due Fee* button on his profile page. Otherwise the PEMS automatically tries once a day.

Andrea Facchini
Antonio Gianola
Andrea Milanta

10. Dealing with traffic fines

Dealing with traffic fines	
Overview	This use case describe how the System manages traffic fines.
Actors	National Transport System
Entry Conditions	The NTS has sent a traffic fine for a specific car at a specific time and location to the PowerEnJoy system.
Flow Of Events	The system looks in the database (generic storage of past reservations) which user was using the car at the time of the infringement. The system charges the user the due fine. (see Charging the user Use Case)
Alternate Flow Of Events	None
Exit Conditions	The fine is charged on the user account
Exceptions	None

Andrea Facchini
Antonio Gianola
Andrea Milanta

11. Emergency alert

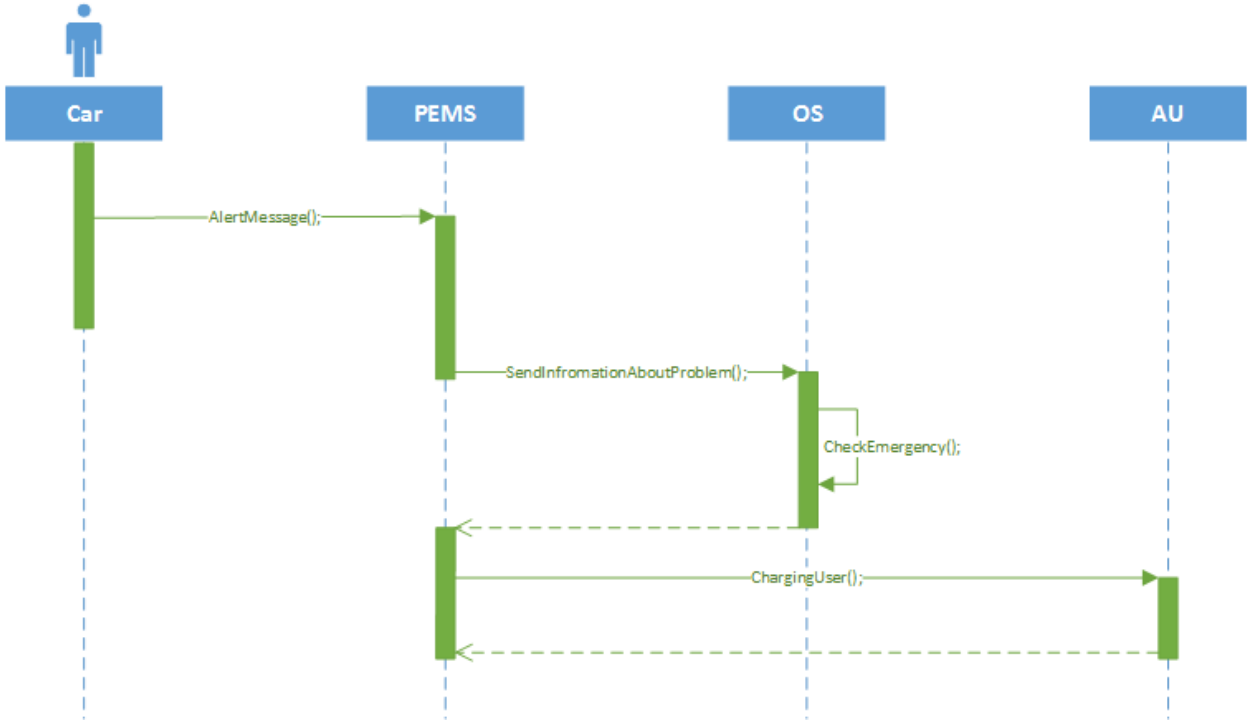
Emergency alert	
Overview	This use case describes how the PEMS manages an exceptional situation.
Actors	Onroad Service, Car
Entry Conditions	An emergency situation has been detected by either the car or the AU
Flow Of Events	The PEMS receives an emergency alert with relevant information from the car (position and type of emergency). The PEMS notifies the OS. The OS verify the proposed type of emergency and notifies the PEMS. The PEMS charges the user extra according to the Emergency Table. The OS resolves the issue.
Alternate Flow Of Events	The PEMS can be notified of the emergency from the OS.
Exit Conditions	The User is charged according to the emergency and the emergency is dealt with.
Exceptions	None

Andrea Facchini
Antonio Gianola
Andrea Milanta

Sequence diagram

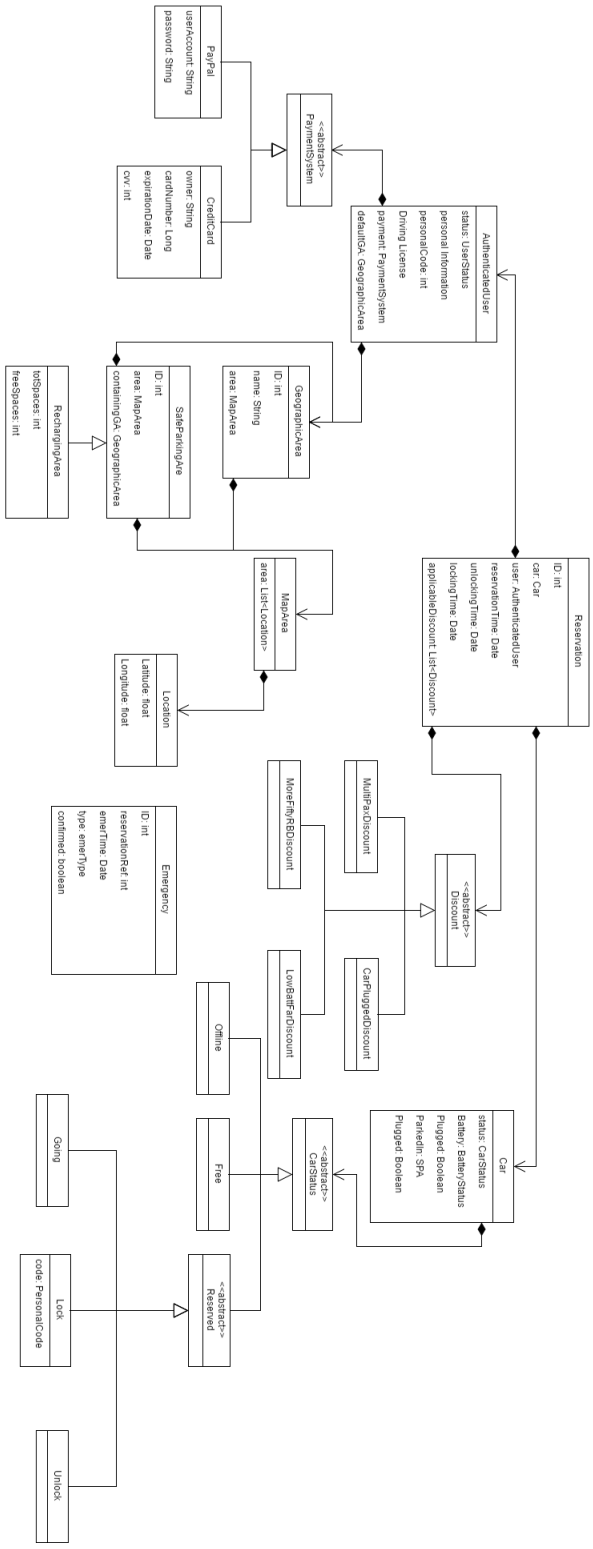
This sequence diagram follows the use cases:

11. Emergency Alert



Andrea Facchini
Antonio Gianola
Andrea Milanta

3.5 Class Diagram

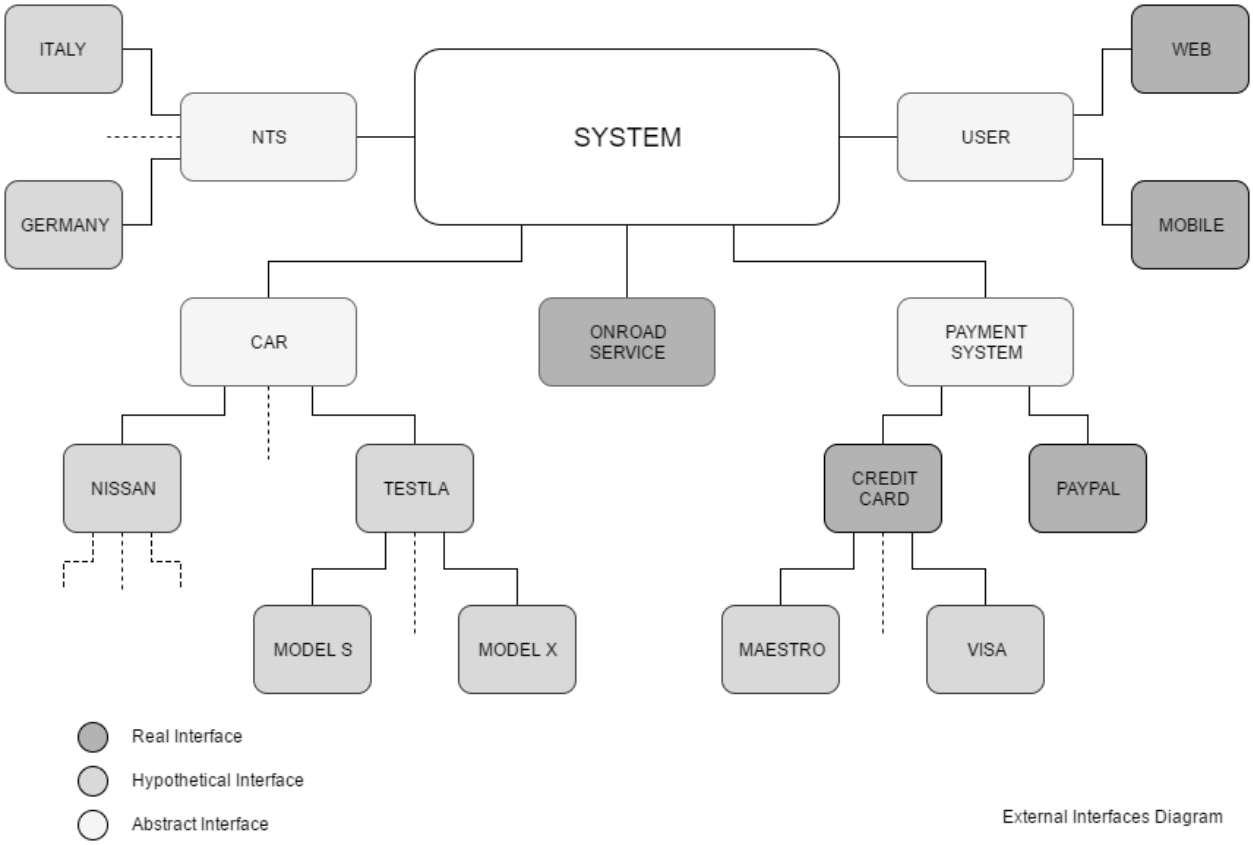


Andrea Facchini
Antonio Gianola
Andrea Milanta

4. Non Functional Requirements

4.1 External Interface Requirements

Due to the nature of the service, the system requires extensive communication with all the different parts involved. Subsequently the proposed system has a complex structure of external interfaces that allows for an easy communication. Such structure has mainly been developed for easy expansion.



External Interfaces Diagram

Andrea Facchini
Antonio Gianola
Andrea Milanta

1. User Interface

1.1. Mobile and Web Application

The main principles of the design are essentiality and usability.
The interface, in both the applications, is structured as essential and linear, marking in a visible way all the possible buttons. This helps also to strengthen the second principle, usability, making the application easy to learn and use.
The main colors are orange (#cc6633), black and white
The main pages of both mobile and web application are shown below.

Login Page

The login page contains all the needed functions in the middle of the screen. These functions are: Login and Registration. The first is immediately accessible, since the 2 required fields are displayed in the same screen, while the second it's a link for the proper registration page (not shown in this document).

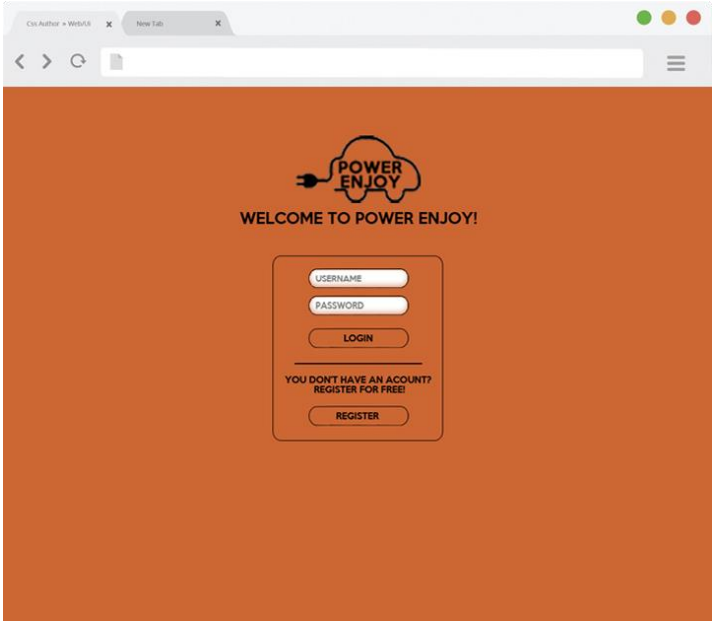


Image 1.1

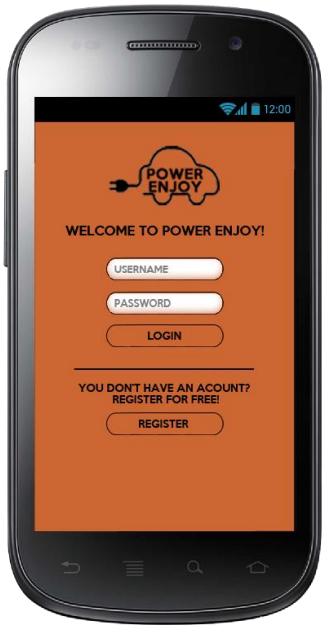


Image 1.2

Andrea Facchini
Antonio Gianola
Andrea Milanta

Main Page

The main page, the one that the AU reach just after the login, it's composed by a main map showing cars and RA, a search bar, for the address, a link to profile information and a link to the setting page. In this screen the AU is allowed to navigate the map in three different ways: using his own GPS position, searching for a specific location or just freely navigate the map with the controls. By touching /clicking on a car or on RA the system displays the information about the selected item.

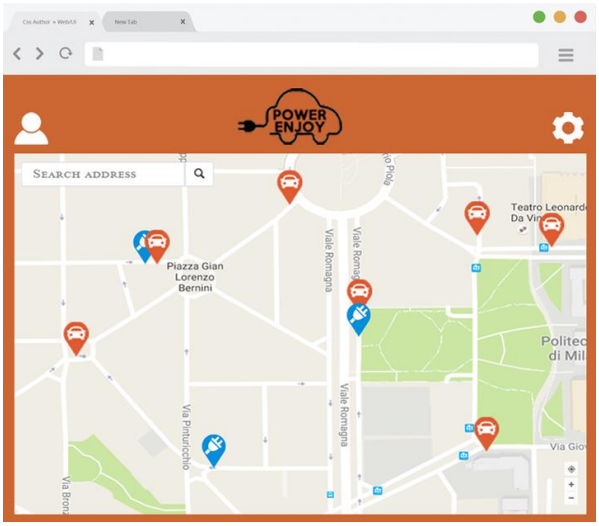


Image 1.3

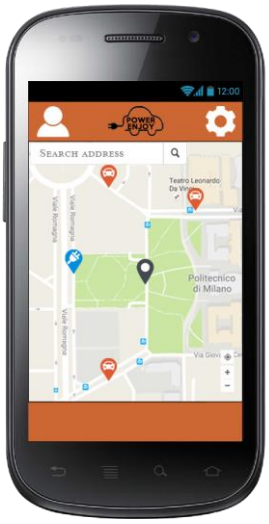


Image 1.4

Car screen

When the AU select a car, the following screen is showing up. It consists in some general information about the vehicle and its residual battery, as well as a reserve and a close button. By clicking on close system starts the reservation procedure and it shows it in the screen.

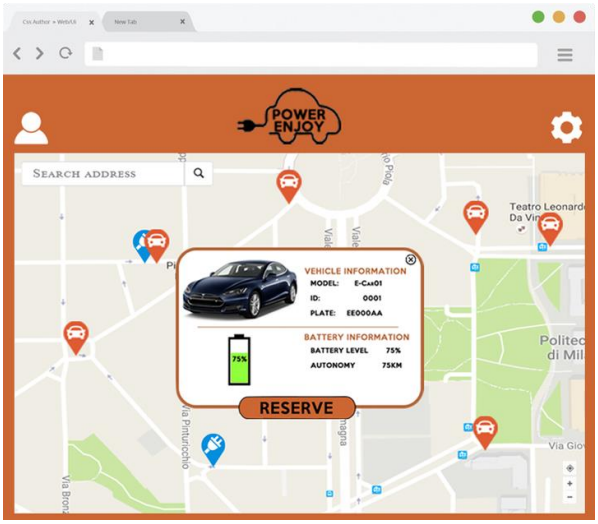


Image 1.5

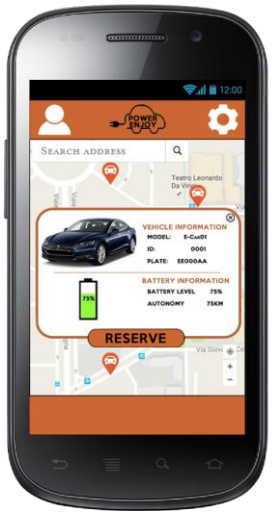


Image 1.6

Reservation screen

When the AU starts the reservation procedure, the screen shows a timer and a cancel reservation button. The timer allows the AU to know how much he have to reach the car he reserved without taking a fine.

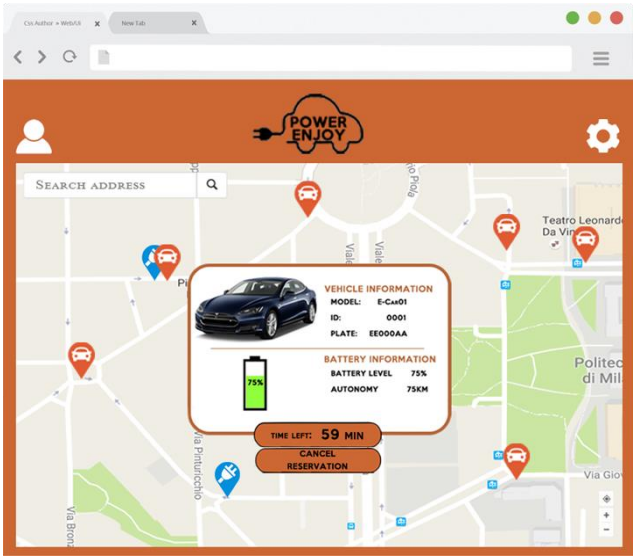


Image 1.7

Image 1.8

1.2. Onboard Computer Application

The interface of the onboard computer of the car maintains the same design of the main application. On all the screen it is possible to identify the white background, with a sidebar on the left and a square for the main function in the middle.
All inputs are given by mean of touchscreen.

PIN screen

In the car, instead of the login, the AU has to insert his personal 4-digit PIN.

Image 1.9

Andrea Facchini
Antonio Gianola
Andrea Milanta

Navigation screen

Once that the AU have authenticated himself with the PIN code, the system shows on the screen the current level of battery, the button for enabling/disabling the MSO and a main search screen for insert the arrival address. The system shows also some shortcuts of similar address inside the SPAs.

image 1.10

Road screen

Once that the destination is entered and the ride starts, the screen shows the battery level, the MSO option, the GPS map navigator and some information about the ride, like the time passed, the estimated time left and the costs so far. Note that this is the base cost, the discount/extra is applied only at the end of the ride.

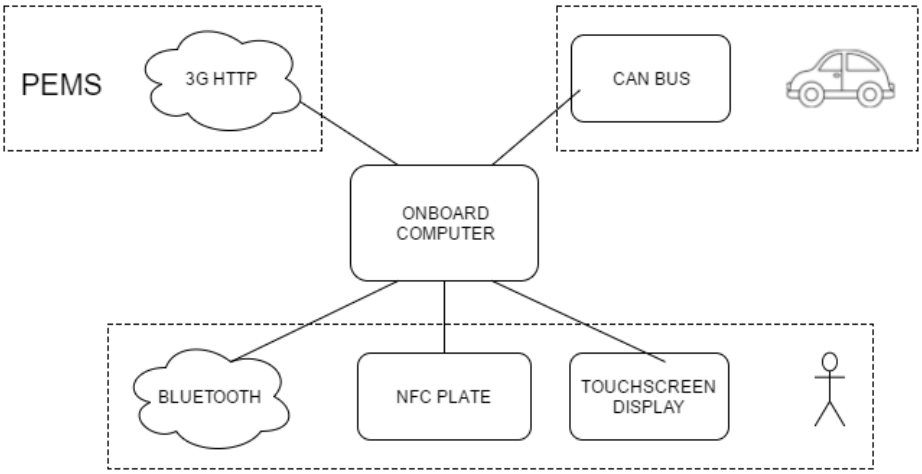
Image 1.11

Andrea Facchini
Antonio Gianola
Andrea Milanta

2. Hardware Interface

2.1. Car Interface

Cars have to be connected to the system in order to be remotely locked and provide the PEMS with the required information.
The cars will be equipped with an onboard computer. This will be able to communicate with the car via a CAN Bus interface, which is the standard communication protocol in automotive. The same computer will provide an interface with the user with a touchscreen display (see User Interface 1.b) and a NFC plate mounted on the windshield. It will also have Bluetooth capabilities to allow unlocking without the NFC.
Communication with the PEMS will be ensured via a standard 3G internet communication.



Andrea Facchini
Antonio Gianola
Andrea Milanta

3. Software Interfaces

3.1. National Transport System Interface

The PEMS will need to communicate with the National Transport System to validate driving licenses.
Although actual implementation may vary depending on the country, the system must be able to retrieve the driving license holder and expiration date using the code, and check whether the license is valid or not.
Note: Traffic fines are received via standard email. A custom implementation for direct communication of fines has been considered but rejected because most of the NTSs are not able to offer this feature yet.

3.2. Payment System Interface

Interface to payment systems is required in order to automatically execute payments. There currently are two main online payment methods.

PayPal

PayPal provides a standard interface for online payment. The PEMS will only need to have the user's credentials.

Credit Card

The most common credit cards systems provide a standard interface that will be used by our system. The PEMS will need all the card information, which will be inserted just once by the user.

3.3. Onroad Service Interface

Interface with the onroad service will happen through a custom built interface.
The PEMS will be able to tell the OS information about a car status and position. In case of emergency it will also tell the type of emergency reported.
At the same time the OS will be able to report to the PEMS emergencies received directly by the user via their call center, and confirm the type of emergency reported.

4.2 Performance Requirement

The system must be able to process any request in less than 20 seconds.
The amount of data shared with user devices must be little enough not to require more than 3% of a standard mobile data plan. (1Gb / month).

4.3 Software System Attributes

The software must have the following attributes

Correctness

The system must be correct to the extent that it satisfies all specifications in this document.

Interoperability

All interactions with remote modules and third-party applications must be monitored and registered.

Portability

The mobile application must be able to run on all the most popular devices.
The web application must be supported by all the most popular browsers.
The onboard computer must be able to interface with all the car models used by the service.

Scalability

The system needs to be easily scalable. Scalability is to be expected in the number of user, but also in the number of cars and GAs.

Reliability

The system must be extremely reliable.
In particular, modules concerning payments and car management (locking, unlocking, …) must be 100% reliable.
Minor bugs are accepted on secondary functions.

Security

Access to the system shall only be granted to authorized users.
Users using the service must be uniquely identifiable.
Users may only see information regarding him or the available cars.
Sensible information, including location, must be securely stored and only accessible from the system and the user involved.

Usability

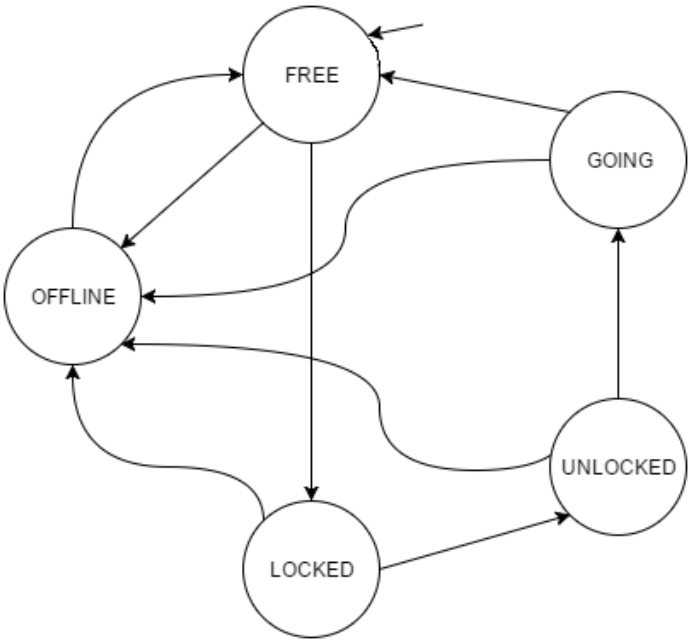
The service provided requires for a simple user interface that allows to perform all the main task easily and fast.
No training is expected from the user.

5. Alloy

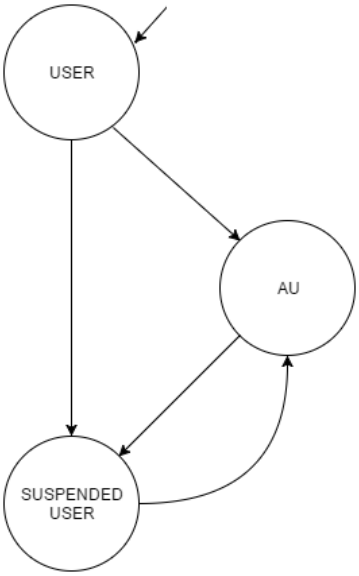
5.1 Model Description

The signatures of the model can be summarized in 4 main fields: Car, User, GA and Reservation. The others serve as field for the main ones or are needed to create connection between them.

- **Car**: every car has a status, a battery level, a presence sensor (for the people onboard), a plug sensor, an engine sensor and a request of MSO. Every car is parked inside a SPA (see the SPA point). The car status can be Offline, Free or Reserved. If Reserved can be Lock, Unlock (doors) or Going. The combination between the status, the presence sensor and the engine sensor identify the correct physical state of the system.
- **User**: a User can be AU or notAU. An AU can be active or suspended. Only active AU have a personal code and can reserve cars.
- **GA**: every RA is contained in a SPA, and every SPA is contained in a GA. To every GA is assigned OS. Every team have a set of car assigned to them. The car is assigned to the team in the current GA when their status is set as Offline.
- **Reservation**: a reservation is linked to an AU and a car (only one of each). It has some parameters needed for calculate the price of the ride, and can have a discount applied, based on the CT.



Car State Diagram



User State Diagram

Andrea Facchini
Antonio Gianola
Andrea Milanta

5.2 Code

```
module PEMS

/*
Sig Part
*/

//The car have a CarStatus
abstract sig CarStatus{}

//the CarStatus can be Offline, Free or Reserved
one sig Offline          extends CarStatus{}
one sig Free extends CarStatus{}
abstract sig Reserved extends CarStatus{}

//if a car is Reserved it can be Lock, Unlock or Going
one sig Lock extends Reserved{}
sig Unlock extends Reserved
{
        code: one PersonalCode
}
one sig Going extends Reserved{}

//boolean sig, needed for some checks
enum boolean{T, F}

//a car can be in charging
enum Charge{isCharging, notCharging}
//BatteryStatus
enum BatteryStatus {Zero, LessThan20, MoreThan50}
//SensorsStatus
enum SensorStatus{NoOne, OnlyOne, TwoOtherMen}


//Car property
sig Car
{
        status: one CarStatus,
        battery: one BatteryStatus,
        plugged: one Charge,
        parkedIn: lone SPA,              //one al più una
        sensor: one SensorStatus,
        MSOrequest: one boolean,
        engineON: one boolean
```


Andrea Facchini
Antonio Gianola
Andrea Milanta

```
}


//All users that access the system
abstract sig User{}

//Code associated to only one User
sig PersonalCode{}

//Authenticated Users (AU) can reserve at most one car
sig AU extends User
{
	status: one UserStatus,
	currentGA: one GA,
	code: one PersonalCode
}

//Non-Authenticated Users can only registers or log in the system
sig NonAU extends User{}

//The AU is Active or Suspended. Superclass UserStatus and 2 extended
enum UserStatus{Active, Suspended}

//The GA is the most larger area
sig GA {}

//like a park or something, is inside a GA
sig SPA {
	inside: one GA
}

//Is a special case of SPA
sig RA extends SPA
{
	totSpaces: one Int,
	usedSpaces: one Int,
}{
	usedSpaces<=totSpaces
}

//Maintenance service
sig Onroad{
	operativeIn: one GA,
	assignedCars: set Car
```


Andrea Facchini
Antonio Gianola
Andrea Milanta

```
}

//Payment and Reservation Manager
sig Date
{
	timestamp: one Int
}{
	timestamp>0
}
enum Discount{NoDiscount, TwoOtherManDiscount, CarPluggedDiscount,
MoreFiftyRBDiscount, LowBattFarDiscount}

sig Reservation
{
	user: one AU,
	car: one Car,
	discount: one Discount,
	reservationTime: one Date,
	unlockingTime: one Date,
	lockingTime: one Date,
}

/*
Pred Part
*/

//dates compare, if a is after b return true
pred DateAfter[a:Date, b:Date]
{
	a.timestamp > b.timestamp
}

//check the reservation property
pred ReservationAU [r: Reservation, u: AU]
{
	r.user = u
}

pred ReservationCar [r: Reservation, c: Car]
{
	r.car = c
}

pred Reserve[c:Car, u:AU]
```

```
{
	some r:Reservation | (ReservationAU[r,u] and ReservationCar[r,c])
}

//car unlocked by the right code
pred UnlockCar [c: Car, u: AU]
{
	c.status.code = u.code
}

//check if the car is in a reserved status
pred IsReserved[c: Car]
{
	c.status = Lock or c.status = Unlock or c.status = Going
}

//check if the car c need assistance
pred needAssistance[c:Car, o:Onroad]
{
	c.status =Offline and c.parkedIn.inside = o.operativeIn
}

//CarParked is true when the car C is parked in the SPA spa
pred CarParked [c: Car, spa: SPA]
{
	 spa = c.parkedIn
}

//check if the ride is over
pred EndRide[c:Car]
{
	(c.status = Going) and (c.engineON = F) and (c.sensor = NoOne)
}

/*
Fact Part
*/

fact ReservationBasic
{
	//for all reservation exist a user and exist a car
	all r:Reservation {
		one u:AU | (ReservationAU[r,u] and u.status = Active)
	}
```


Andrea Facchini
Antonio Gianola
Andrea Milanta

```
        all r:Reservation {
                one c:Car| (ReservationCar[r,c])
        }
        //no car has two reservation = No reservations have two cars
        all r1,r2:Reservation | (r1 !=r2) => (r1.car != r2.car)
        //no user has two reservation = No reservations have two AU
        all r1,r2:Reservation | (r1 !=r2) => (r1.user != r2.user)
}

fact BatteryProperty
{
        //All cars with battery = Zero have status = Offline
        all c:Car | (c.battery = Zero) => (c.status = Offline)
}

//manage the car status of the system.
fact CarStatus
{
        //All reserved cars are in a valid state
        all r:Reservation | IsReserved[r.car]
        //all car not in Reservation Set are free or Offline
        (Car - Reservation.car).status = Free or (Car - Reservation.car).status = Offline
}

fact ReservationProperty{
        //for all car if car is in Lock, Unlock or going status exist one and only one reservation for
this car
        all c:Car | ((IsReserved[c]) <=> (one r:Reservation | ReservationCar[r,c]))
        //if C is offline or free, can not be reserved
        all c:Car | (c.status = Unlock) => (one u:AU | UnlockCar[c,u] and Reserve[c,u])
}

fact ReservationNumber
{
        //there are no two cars with the same unlock
        all c1,c2:Car{
                        all ul1,ul2:Unlock | (c1 != c2 and c1.status = ul1 and c2.status = ul2) => (ul1
!= ul2)
                }
        //if the car is reserved for one persone this car has the same code of the person
        all c:Car, u:AU | (Reserve[c,u] and c.status = Unlock) => (UnlockCar[c,u])
        //Per ogni utente u1 per ogni utente u2. u1 != u2 sse i 2 codici sono diversi
        all u1, u2:AU | (u1 !=u2) => (u1.code != u2.code)
}
```

```
//for a better visualization
fact NoAloneProperty
{
	//no alone code
	all c:PersonalCode {
		some u:AU | u.code=c
	}
	//no date alone
	all d:Date {
		some r:Reservation | r.reservationTime=d or r.unlockingTime =d or
r.lockingTime=d
	}
}

fact UnlockProperty
{
	//Each Unlock is assigned to a Car
	all ul:Unlock | (one c:Car | ul = c.status)
	//Each different unlock have different Code
	all ul1,ul2:Unlock | ul1!=ul2 => ul1.code !=ul2.code

}

fact AUIsInTheCar
{
	//If car is Going or Unlock Sensor is != NoOne
	all c:Car | (c.status = Going or c.status = Unlock) => c.sensor != NoOne
	all c:Car | (c.status = Lock or c.status = Free or c.status = Offline) => c.sensor = NoOne
}

fact ParkedProperty
{
	//if car is free, Lock or Unlock exist one and only one SPA in ParkedIn
	all c:Car | (c.status = Free or c.status = Lock or c.status = Unlock or c.status = Offline) =>
(one spa:SPA | CarParked[c,spa])
	//if a car is in Charging it is in RA
	all c:Car | (c.plugged = isCharging) => (c.parkedIn = RA)
	//if a car is Charging it cannot move
	all c:Car | (c.plugged = isCharging) => (c.status != Going or c.status != Offline)
	//# of car in a RA is <= max space (come si fa????)
	//If a car is Going it is no parked
	all c:Car | (c.status = Going) => (no spa:SPA | CarParked[c,spa])
```

```
}

fact Discounts
{
	//se il motore è spenso non c'è discount
	all r:Reservation | (r.car.engineON = T and r.car.sensor = TwoOtherMen) <=> r.discount
= TwoOtherManDiscount
	//se la corsa è finita e la batteria è più di 50 allora c'è lo sconto
	all r:Reservation | (EndRide[r.car] and r.car.battery = MoreThan50) <=> r.discount =
MoreFiftyRBDiscount
	//se la corsa è finita e la batteria è meno del 20 allora extrapayment
	all r:Reservation | (EndRide[r.car] and r.car.battery = LessThan20) <=> r.discount =
LowBattFarDiscount
	//se la macchina è plugged allora sconto
	all r:Reservation | (EndRide[r.car] and r.car.plugged = isCharging) <=> r.discount =
CarPluggedDiscount
}

fact GoingProperty
{
	//all car not going are really not going (engine off)
	all c:Car| (c.status != Going )=> c.engineON= F
}

fact MSOProperty
{
	//no  one can ask for MSO or have MSO when not going
	all c:Car | (c.status != Going => (c.MSOrequest = F))
	//some who ask for MSO after they verify it
	some c:Car | (c.status = Going => (c.MSOrequest = T))
}

fact RoadService
{
	//all GA have a service
	all g:GA | one o:Onroad | o.operativeIn = g
	//offline cars are taken care by the service in the same GA
	all c:Car {
		all o:Onroad |(needAssistance[c,o]<=> (c in o.assignedCars))
	}
}

/*
Assertion Part
```

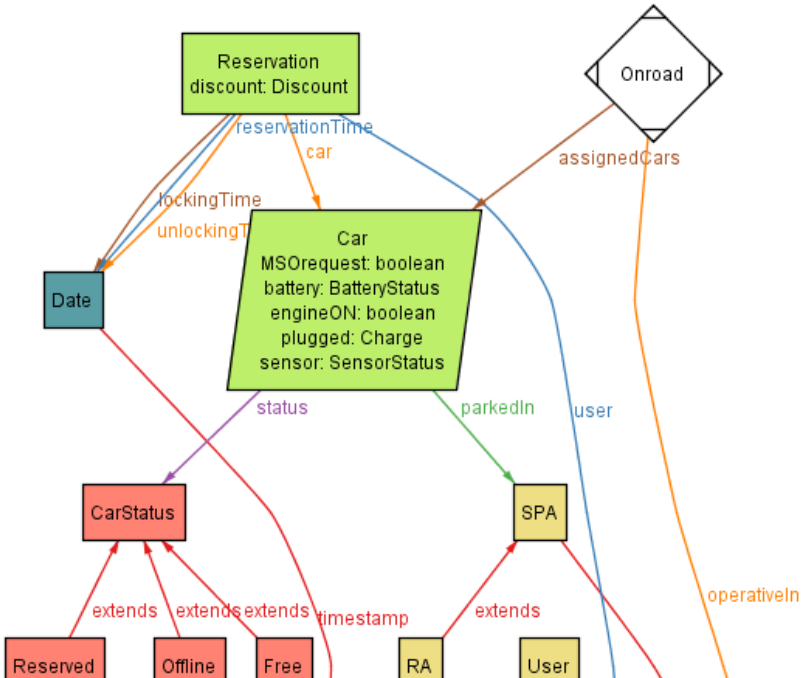
```
*/

assert ReservationCar
{
	//For all reserved cars the owner is not suspended
	all u:AU, c:Car | Reserve[c,u] => !(u.status = Suspended)
	//no car has two reservation -> it is a fact, is useless
	all u1:AU, u2:AU, c1:Car, c2:Car | (Reserve[c1,u1] and Reserve[c2,u2] and u1 != u2) =>
c1 != c2
	//for all free or Offline car there is no reservation
	all c:Car | (c.status = Free or c.status = Offline)=> (all u:AU | !Reserve[c,u])
	//not exist a reserved car without reserver
	all c:Car | (c.status = Reserved) => (one u:AU | Reserve[c,u])
}

assert SPAProperty
{
	//All SPA are in one and only one GA:
	all spa1:SPA{
			all spa2:SPA | (spa1=spa2) => (spa1.inside = spa2.inside)
	}
	//se una macchina è parked allora ParkedIn = SPA
	all c:Car | (c.status = Free) => (one  spa:SPA | CarParked[c,spa])
	//If car is charging is in RA
	all c:Car | (c.plugged = isCharging) => (c.parkedIn = RA)
}

assert UniqueCode
{
	//there are no two different AUs with the same code
	no u1:AU, u2:AU | u1 != u2 and u1.code = u2.code
}

assert BatteryCheck
{
	all c:Car | (c.battery = Zero => c.status = Offline)
}

//check some constraints
assert Integrity
{
	//reservation integrity
	no r:Reservation|(#r.user>1 and #r.car>1)
	//car integrity
```

```
        no c:Car|#c.status>1
        //onroad integrity
        all c:Car|c.status = Offline => one o:Onroad| c in o.assignedCars
}



//show a feasible scenario
pred A
{}

run A for 4

//check assertion
check ReservationCar for 4
check SPAProperty for 4
check UniqueCode for 4
check BatteryCheck for 4
check Integrity for 4
```

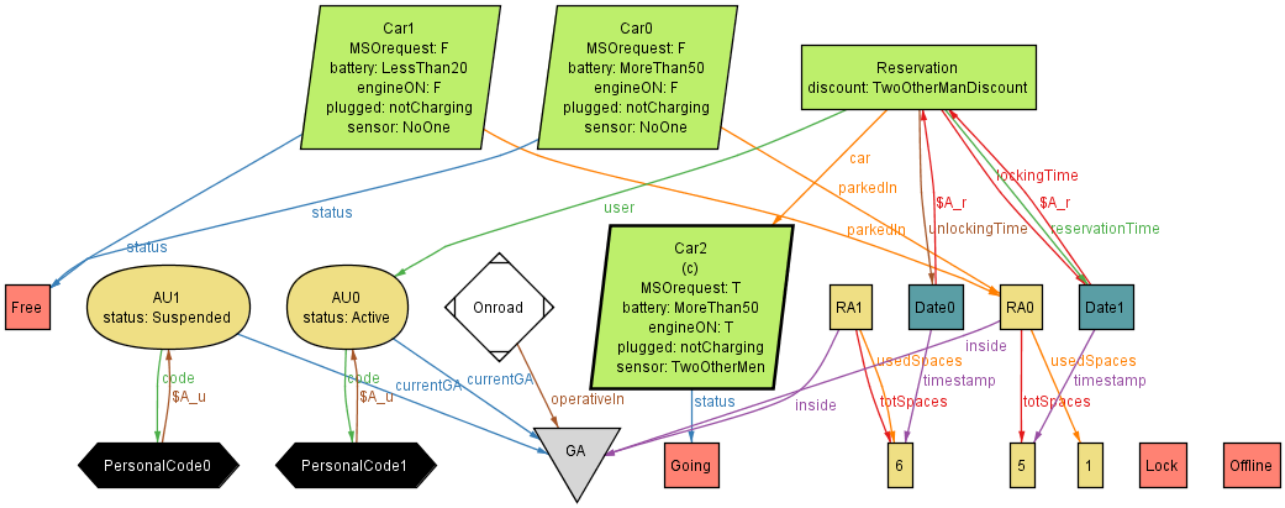
5.3 Result

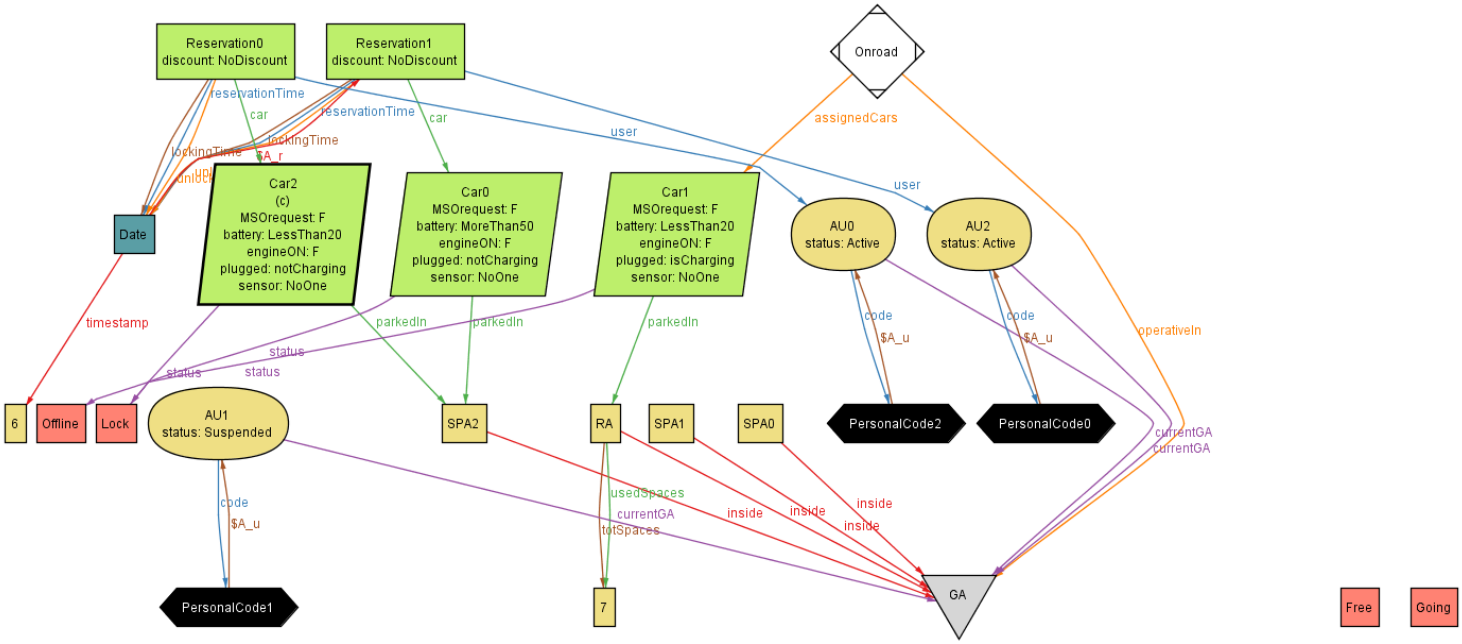
6 commands were executed. The results are:
#1: **Instance found.** A is consistent.
#2: No counterexample found. ReservationCar may be valid.
#3: No counterexample found. SPAProperty may be valid.
#4: No counterexample found. UniqueCode may be valid.
#5: No counterexample found. BatteryCheck may be valid.
#6: No counterexample found. Integrity may be valid.

5.4 Model Graph



5.5 World Instances





Andrea Facchini
Antonio Gianola
Andrea Milanta

6. Appendices

6.1 Tables

1. Cost Table

Code	Price Variation	Requirements
D01	10%	At least 2 passengers on the car.
D02	20%	RB >50%.
D03	30%	The user let the vehicle plugged in a RA.
F01	30%	RB<20% and Nearest RA at more than 3 KM.

2. Emergency Table

Code	Price	Description
E01	250€	Accident
E02	250€	Empty Battery
E03	250€	Car Left Out of SPA
E04	0€	Technical Malfunction