PowerEnJoy

Requirement Analysis and Specification Document

Cattaneo Davide El Hariry Matteo Frontino Francesco

Contents

1 Introduction

1.1 Purpose

This project aims at designing an electric-car sharing software system.

Car Sharing is a very cost-effective and useful service for anyone who needs a car occasionally. It allows people to use and pay for the car according to their personal use, without the hassle and costs of owning their own vehicle (parking, purchase costs, maintenance, insurance etc.).

The system we will develop is meant for cities which are provided with an efficient amount of parking lots and a wide distribution of electric car-charging platforms throughout the urban areas.

The application must allow the users which are registered to perform several easy and effective operations. Once logged in, the user can find available cars around him/her or in specified locations of the city, and chose the one to reserve.

Afterwards the user, who needs to reach the car before a given time slot expiration, will be able, by unlocking the car using the app, to easily enter the vehicle and drive to his/her destination.

1.2 Goals

To create a throughout solution for the problem of providing a good and effective service to our users, the application must allow the following:

- [G1] Users must be able to register and access to the system.
- [G2] Registered users must be able to find the locations of available cars within a certain distance.
- [G3] Users must be able to reserve a single car with a one-hour time limit.

•[G4] If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; user will be charged with a fee of 1 €.

•[G5] A user that has reserved successfully a car must be able to unlock it using the app.

•[G5+] Once a user is inside the vehicle (s)he must be able to indicate eventual damages in the car.

•[G6] As soon as the engine ignites, the system starts charging the user for a given amount of money per minute; the user is notified of the current charges through a screen on the car.

•[G7] The system stops charging the user as soon as the car is parked in a safe area and the user exits the car; at this point, the system locks the car automatically.

•[G7+] The cars parked not in safe area must keep charging with the halt rate up to a given limit of time Max-Time-Stop.

•[G8] The set of safe areas for parking cars is pre-defined by the management system.

•[G8+] The set of safe areas must always be displayed on the car display's map during the rides.

•[G9] system must know parking location, the battery level, status (in charge or not) and if there were two passengers onboard every time a ride is over in order to calculate the right discount.

•[G10] The system must charge the proper cost for every user at the end of the ride, so that eventual discounts or fee are included.

•[G11] Mechanical problems that occur during a ride session and accidents must be solved by the maintenance operators, whom can be contacted by users through the car display.

1.3 Definitions, acronyms and abbreviations

Here is a brief description of the most important actors and words used in our system:

- **User:** by user is meant a person already registered in the system, so that has a profile, uses the features provided by the system and performs actions accordingly. (S)He can use all the functionalities described below (see Functional Requirements).
- **Guest:** a guest is a person that probably for the first time accesses the system or that hasn't already signed up. Guest has less power in the system; his/her actions are limited to access an introduction view and register to the service.
- **System:** is the application core. The software system which will perform all the operations and monitor interactions and be a medium between users and cars.
- **Reservation:** the allocation of a car to a user, which starts when the booking request arrives and ends either when the expiration time ends or when the car is unlocked. In this last case it triggers the start of the first travel so it initiates a ride.
- **Car:** the vehicle used by the users, which contains different sensors and an embedded computer. It has seat sensors to detect passengers, sensor to know battery level and charging actions. The computer, of course, has as main functionality to provide navigation facilities through a GPS system and to send all the relevant data to the main system server.
- **Ride:** conceptually is the use of the car, and it can be identified by the time duration of the user's journey, from unlocking the vehicle until the final parking (having user selecting "end ride" or "end ride & charge" on the car screen) with the car locked.
- **Travel:** is considered as the ride segment and is identified by a change of the status of the car. More travels can be part of a single ride.

- **Operator:** is a flexible actor in our system. He's part of a set of people operating under the administrator directions. Their normal tasks are to bring to charging stations cars left with less than 15% battery level, interact with users which call for help during a ride, intervene when necessary (e.g. a wheel brakes during a ride). Their exceptional task can be the case in which they have to go and get back cars taken by the police or cars involved in incidents etc.
- **Administrator:** the administrator of the system is the person allowed to manage eventual unexpected cases (like incidents and damaging situations). He is the person notified every time a problem occurs, and once analyzed the situation (s)he'll decide how to handle it (call for support, send operators, call the police etc.).
- **Safe Area:** is a part of a set of areas considered safe for parking cars after a ride is over. Temporary stops can be everywhere, but long term parks can only occur in safe areas. They must be very spread and every neighborhood should have at least one.
- **Normal rate:** the charging rate applied when the car engine is ON.
- **Halt rate:** the charging rate applied when the engine is OFF and either the user is inside or (s)he has parked the car in temporary stop mode.

1.4 Reference documents

Specification Document:
* Assignments 1 and 2 (RASD and DD).pdf
* IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

Examples documents:
* RASD sample from Oct. 20 lecture

2 Overall description

2.1 Product perspective

The proposed solution consists of a mobile application platform that will provide the public with the services described below.
Users will be able to reserve the car which better suites their location preference and enter it as soon as they unlock it using the app.
Once the car is unlocked the system gives the user 1 minute to enter it before locking the car again. Once user is inside he must, by interacting with the touch-screen, signal eventual damages at the car, if there are any. From when the user is inside the vehicle the system start charging him/her with the halt-rate until (s)he starts the engine (then the normal rate starts). The rate will keep changing according to the engine mode (normal rate when engine is ON, halt rate when OFF) until the user reaches his/her destination and permanently exits the vehicle by choosing to end the ride. User may also interact with the system by handling his/her own profile with different provided operations (check previous rides, check personal stats etc.).
The administrator will be given a special module to manage specific and special situations.

2.2 Domain properties

It is supposed that the following conditions hold in the analyzed world.

- The system must be able to store user information.
- GPS always give the right position.
- Cars shown to users trying to make a reservation are always and only the available ones.
- Available cars always have an active GPS.

- Two reservations for a single car can't occur on the exact same time.
- A user can reserve only one car at a time.
- A user can only make a new reservation if there isn't one pending on him.
- After a reservation expires the user is allowed to make a new one.
- Reservation processes are addressed as time limited transactions.
- A booked car will always be found in the location provided at the reservation time within one hour.
- The system must start the time counter when a booking confirmation is received.
- Available cars can only have esthetical damages, that don't compromise in anyways the usage of the vehicle according to our car-sharing system.
- Mechanic damages are considered all those problems the car may have which will negatively affect the driving and car usage in general.
- Mechanic damages can be electronically detected by our cars and though signalled to the main system in real time.
- The only way to unlock the car is by means of the smartphone app.
- The user has 1 minute of time to enter the car before it locks again.
- All parked and not in use cars are locked.
- A car must be unlocked only if it is in use or if an authentication request is processed as successful.
- The car system is able, through the seat sensors, to know when a person is inside the vehicle.
- When a user enters a car the engine must be off.
- When a user enters a car the screen must be working.
- When the start button is pressed the car must turn on the engine.
- The charging fee unit is money per minute.
- The car system is always able to send information through mobile network to the main system.
- Mobile networks are always providing a fair signal to all devices and computers of our system.
- User's money is always available due to the use of credit cards as the only mean of payment granted.
- It is not possible for the user to put in charge the vehicle during a temporary stop.

2.3 Assumptions and rationales

There below is specified how the world is assumed to work in ambiguous situations:

- The car is unlocked when the user, after reserving the vehicle, uses the app function "unlock car". This action will grant access to the associated car for up to 1 minute, time when the system locks the car again if nobody is detected inside.
- Users have the responsibility of esthetical damages, so it is their duty to signal if the car they reserved has any, otherwise the guilt of those eventual problems will automatically be charged on them.
- Once they have opened the car, users might want to signal some damages they found, so they have to simply interact with the proper interface. If no damage is reported but actually something is wrong, the cost of repairs will be charged to the last user who didn't care about informing the system about the damage.
- During the period between the car opening and the car locking, in which the car has persons inside but the engine is not, the user will be charged with a halt rate, which will be lower than the normal charge but useful in avoiding people from occupying the vehicle for free without using it.
- If during the driving some problem that concerns the car or its usage occurs, users can call for help using the apposite button provided by the car screen. Once confirmed the help request, an operator will be put in charge of the situation and (s)he'll manage the situation.
- The system is provided with a list of safe parking areas.
- When dealing with safe parking evaluation, we assume that the onboard computer can identify each time if the parking has a match among the safe ones on its list.
- The information needed to estimate the discount for each ride, will be available to the system for each car. The system can get the battery level/status, car location and passengers number through sensors mounted in the vehicle. Then, knowing those parameters, it will calculate the eventual discount for the user.
- The discount will be applied to the user's account only once the ride is finished and the car locked. This because the system needs to know whether the car after the usage has been plugged to the power grid or not and the exact location where it has been left.

- If the user has not plugged the car in charge within 1 minute from the selection of the option “end ride and recharge” through the car display, the system is discarding that option, locking the car and applying the “end ride” action only.
- System informs users about the possibility of getting discounts and about how to obtain cost reductions. All the information about discounts and car usage are written in an appropriate module openable from the app menu or accessible from the web site.

More domain assumptions:

- The bonus related to the battery savings is to be applied at the end of a ride, after users confirm the ride is over. The eventual 20% discount is t o be applied to the total cost of the entire ride.
- The 50% of the battery is considered dynamically on the total battery le vel available at time of booking. If one takes a car with low battery is b ecause (s)he expects to drive for a short time, so it is normal that it is m ore difficult for him to get the discount.
- The 30 % fee is to be applied considering that the battery has to be left with more than 80% empty level of charge with respect to the charge le vel that the car had at the reservation time.
- A car can only be taken if the battery level is higher than 15% of the tot al charge.
- Under 15%, in case the car is not connected to a charging station, the s ystem will notify an employee to go to recharge the car, possibly movin g it.
- A ride can be composed by several travels.
- Travels are considered as the ride segment between two changes of stat us.
- The 2 passengers related bonus is calculated for each travel according t o every single travel charge for that ride.
- The 2 passengers related bonus is calculated on the cost of each travel a nd it is assigned at the moment when the doors are closed, engine is ru nning, and the seat sensors detect at least three passengers onboard (o ne is the driver and at least two passengers).
- The car, for security reasons, turns on and remains on as long as all the doors are properly closed.
- The passenger capacity is calculated by weight sensors placed under th e seats which detects a mass greater than 20 kg
- For every user, the system stores a list of his travels: among the variou s information for a single ride, the system saves an itinerary compose

d by a set of positions. Thanks to this routes information, it is able to h andle infractions, road violations, fines etc. so the costs of the eventual fines and law breaking behaviors can be assigned to the right user.
- Fines received by users will only be forwarded to them, including a cop y of the itinerary information. It is then up to the user to decide whethe r to pay the fine or otherwise. PowerEnJoy only duty is to inform the la w enforcement by providing user data to them at the time of the infring ement.
- Car charges are handled in this way: when the user finishes a ride, the car is able to detect, through the system, its location within a charging station. If the user chooses the option "park & charge", the car queries t he system which in turn sends the vehicle the number of the column to be used, enabling it for use within 1 minute. Columns not enabled do n ot deliver current.
- Areas are mapped as circles with center and diameter
- One car can be disconnected from the charging column by an operator o r by a user who has booked it and unlocked it.
- Any damage that occurs the cars parked and plugged in charge are alw ays related to the user who has placed in charge the vehicle. The rechar ge areas are under video surveillance to prevent vandalism from being charged as a user fault.

2.4 Possible future implementations

- The service could be extended to disabled people by adding special vehicles to the company's set of cars.
- The service could be extended to electric motor-bikes.
- The service could allow the exchange of vehicles between cities in which PowerEnJoy operates.
- The possibility to pay with other methods than the credit card could be added to the system (methods like debit card or phone credit for example).

3 Specific requirements

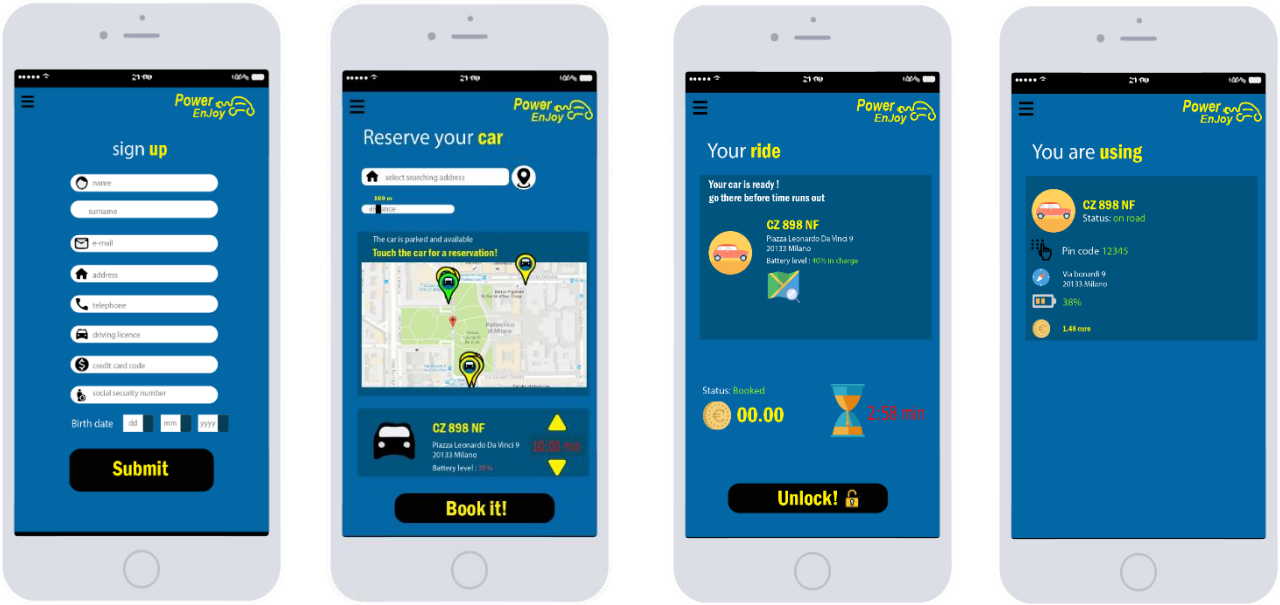
3.1 External interface requirements

Here are provided further details on how the system is intended to interact with the user

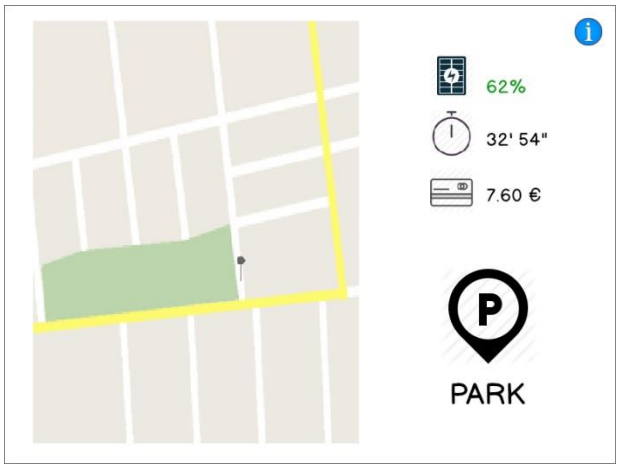
3.1.1 User interfaces

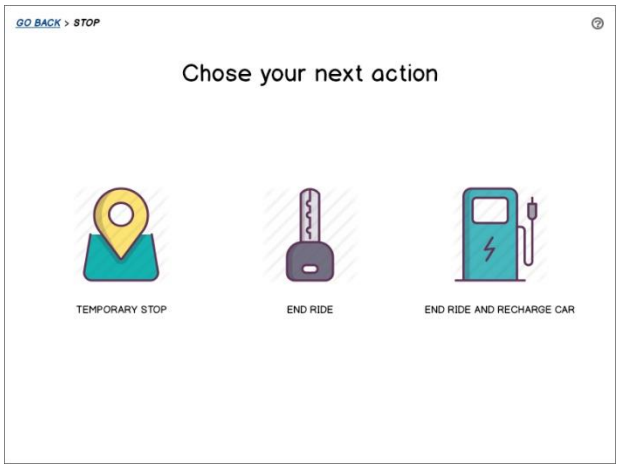
User interfaces reported below should give a general idea concerning how functionalities are provided and how the system interaction has been thought. Mobile app and car system interfaces are presented.

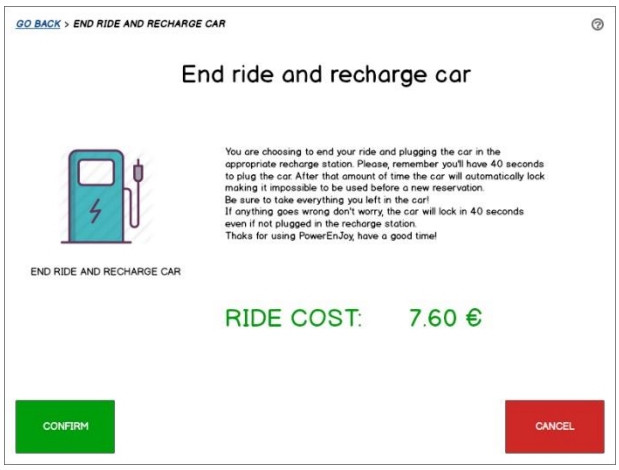
3.1.1.1 Mobile interface



3.1.1.2 Car interface







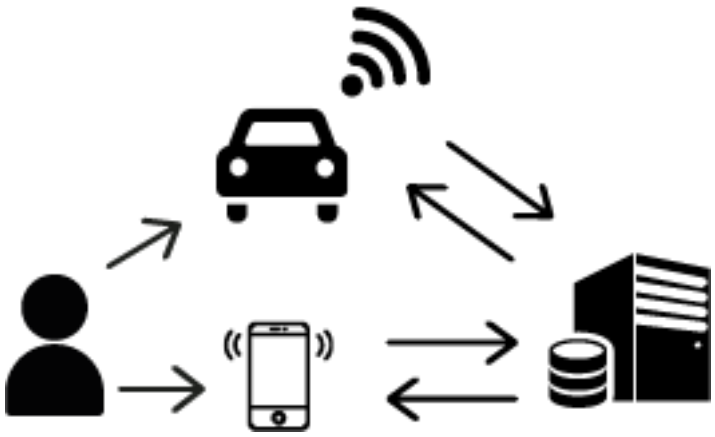
3.1.2 Hardware interfaces

The car sharing service is made available only to those users which are in possess of a modern smartphone or tablet and satisfy the constraints expressed in chapter 3.1.3. No other physical device is permitted to unlock a car.

Each car has an embedded micro-computer aiming to assist the user and receive commands throughout a touch display.

Moreover, all vehicles are provided with a set of functioning sensors (seat weight sensors, GPS, door sensors and charge-plug sensor) useful to grant an efficient check of car statuses and conditions.

Each car is provided with a mobile communication technology platform allowing it to be permanently connected and reachable through the Internet. This connection is used to exchange data between the car and the system.



3.1.3 Software interfaces

The user must be in possess of the official PowerEnJoy app. The app is available to download on Google Play, Apple Store and Windows Store for every mobile device having respectively:

- Android 4.4 or above releases
- iOS 7.0 or above releases
- Windows Mobile 10

Accepting usage condition is essential to use the app. No other 3rd party software is officially supported nor allowed.

To access the PowerEnJoy site, a web browser supporting HTML 5/CSS 3 is needed.

3.1.4 Communications interfaces

Devices used to unlock and reserve a car must be connected to a telecommunication network. Thus, at least one of the following technologies must be supported:

- 3G
- HSDPA
- 4G

Wi-fi networks can also be used to unlock a car but a mobile communication modem is needed to download the app.

Devices which are not provided with at least one of the listed communication technologies cannot be used in any way to reserve and unlock a car.

Each car is provided with a mobile communication technology. Connection signal is guaranteed to be available 24/7.

3.2 Functional requirements

- **[G1]** Users must be able to register and access to the system:
  - system must be able to provide a registration form, in which the user must enter his/her personal data and payment information in order to successfully register.
  - the correctness of credit card information is evaluated by a specific external bank modules linked to our system.
  - system must automatically verify the social security number and the email validity and eventually send the user a password, which (s)he can use to log in.

• [G2] Registered users must be able to find the location of available cars within a certain distance.

- system must be able to find all available cars within the distance defined by the user. This functionality is provided by the application itself.
- user can choose the searching position from his own location (using the phone GPS) or from an address (s)he manually inserts.
- system must be able to indicate the vehicles in the perimeter range on the map.
- for each indicated car on the map system will show its battery level.
- system will show on the map only cars which are available in the searching region defined by the user inputs.
- system must hide a reserved car from the map until its release.

• [G3] Users must be able to reserve a single car with a one-hour time limit.

- user must be able to reserve an available car shown by the system, by selecting it directly from the map or a list, provided by the application, and choosing "reserve car" option.
- reservation lasts one hour.
- system must hide a reserved car from the map until its booking expires.

• [G4] If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; user will be charged with a fee of 1 €.

- a car must be released from a reservation when the one-hour slot of time expires. This is done by changing its status from "reserved" to "available".
- a car release implies its reinsertion among the available ones.
- system must charge the user who books a car and lets the reservation expire with a fee.

•[G5] A user that has reserved successfully a car must be able to unlock it using the app.

- the car system must be able to remotely unlock the car that's been reserved when the user uses the app function "unlock car".
- the car stays unlocked for up to 1 minute.
- the system locks the car after 1 minute from the user "unlock request" if (s)he does not enter in that period of time.

•[G5+] Once a user is inside the vehicle (s)he must be able to indicate eventual damages in the car.

- the display, once the user is detected inside the vehicle, must show the user a view that allows him/her to indicate esthetical damages to the car.
- if the user points out one or more damages, the system admin will be notified and (s)he'll set the vehicle as in maintenance right after this user's ride.
- maintenance operators will take care of the damaged cars and prior user to the damage indication will be charged the costs of repairs.

•[G6] As soon as the user unlocks the vehicle, the system starts charging him/ her for a given amount of money per minute; the user is notified of the current charges through a screen on the car.

- the electric car engine ignites when the user presses the "start" button in the car dashboard.
- the car system must start charging the user when he/she starts the engine.
- the charging fee must be shown on the car screen and kept updated throughout all the driving time.

•[G7] The system stops charging the user as soon as the car is parked in a safe area and the user exits the car permanently; at this point, the system locks the car automatically.

- the set of safe areas for parking must be defined before the system is being used for the first time.
- the car system, thanks to seat sensors, can determine when the user has left the car.
- once the previous conditions are applied, the system stops charging the user and starts calculating the eventual discount/fee (see *advanced functional requirements)*.
- if the car is not parked in a safe area, the car can only be parked in temporary mode, so the halting rate will start charging (see *goal [G7+]*).

•[G7+] The cars parked not in safe area must keep charging with the halt rate up to a given limit of time Max-Time-Stop.

- Cars not in safe areas can only be parked in temporary stop.
- When a user leaves the car in temporary stop, will be notified about the limit of time.
- If the limit of time is exceeded and user does not return an operator will take care of taking the vehicle in a safe area and set the car as available again. The user will be charged the bill up to the expiration time for allowed stop.

•[G8] The set of safe areas for parking cars is pre-defined by the management system.

- the system must be flexible in order to adapt to the city in which it is deployed. In fact, it allows the administrator to set the number and location of safe areas for parking and to modify them.

• [G8+] The set of safe areas must always be displayed on the car display's map during the rides.

- the car display must always show the map.
- the map must be built so that it shows clearly the areas where the car can safely be parked.
- this is meant to help users finding places where to leave the cars without any problem.

• [G9] System must know parking location, the battery level, status (in charge or not) and if there were two passengers on-board every time a ride is over in order to calculate the right discount.

- in order for the system to evaluate properly an eventual discount for the user's ride, it has to know the parking location, the battery level, status (in charge or not) and if there were two passengers on-board.

• [G10] The system must charge the proper cost for every user at the end of the ride, so that eventual discounts or fee are included.

- the system will quickly and accurately calculate the cost of each ride summing up all the travels cost and eventual discount or fees.
- the ride costs are charged on the users bank account thanks to their credit card information given during registration.

• [G11] Mechanical problems that occur during a ride session and accidents must be solved by the maintenance operators, whom can be contacted by users through the car display.

- Mechanical problems are all the car issues that won't allow the vehicle to be used properly or to be driven (e.g. broken wheels etc.)
- When some problem occurs users must be able to easily contact the maintenance through the car-display option "call for help".
- Once the call is received by administrators or operators themselves, they can talk with users in order to know the entity of the problem and decide how to handle the situation.

Advanced Functional requirements

a) If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.

D: the car is provided with always functioning seat sensors.

- seat sensors must be able to detect correctly a person on-board and notify the car system.
- the car system must correctly evaluate the seat sensors notifications and eventually report a 10% discount on the current ride.

b) If a car is left with no more than 50% of the battery empty (dynamically calculated on the initial battery level of the ride), the system applies a discount of 20% on the last ride.

D: battery level sensors always work properly on each available car of the system.

- battery sensors must be able to notify the car system about the battery level (in a percentage scale) in real time.
- the car system must correctly evaluate the battery level notifications when the drive is over and eventually report a 20% discount on the current ride.

c) If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.

- special parking areas are predefined and stored in the database of the system.
- all special parking areas have plugs to charge the electric cars.
- battery plug sensor always work properly on each available car of the system.
- battery plug sensor must be able to notify the car system about the battery status (plugged or not) when the car is parked.

- system must be able to match if the parking where a car is located is part of the special parking area.
- battery plug sensor must be able to notify the car system about the battery status (plugged or not) when the car is parked.
- from the time a user ends a ride the system will detect for 1 minute if the vehicle is being plugged in charge, and in case it is not, it will lock the car and apply no discount.
- car system must apply a 30% discount to the user when the car is parked in a safe area and it is plugged in charge. When it is plugged to the power grid without being in a safe area no discount is applied, but if the user didn't end the ride (so (s)he is in a temporary stop) the system will stop charging him/her any rate.

d) If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty (dynamically calculated on the initial battery level of the ride), the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.
   - the car system must find the nearest power station to the cars which right after a ride are parked and are not connected to a charge grid. When the distance found is greater than three kilometres, or when in the same case the parked car is not being plugged and the battery is less than 20% level, the user must be charged 30% more on the last ride.

e) If the user enables the money saving option, (s)he can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

   D: Money saving option can only be activated if battery level is below 90% of charge.

   - the option will only be displayed on the car screen, so be selectable, if the battery level is less than 90% of the total charge. This to ensure that cars which are quite full, so don't actually need to be immediately charged, will not occupy charging plugs in the available stations.

- the system keep track along with the location of cars (distributed throughout the city) of their status and their battery status and level (in-charge).
- the cars provide a selectable ride option called “money saving”. This option is displayed by the car screen after the user enters the car. The user can start the saving mode in anytime from the driving start till the stop. The option allows the user to find the nearest charge station from the current location or from his/her destination. The charge stations which best suits the searching parameters (location and distribution of other vehicles in charge) will be indicated in the navigation map on the screen.
- the money saving option uses a special algorithm able to take into account the availability of power plugs at the nearest station at the searching time, in order to provide an accurate suggestion. The algorithm also, using the information about other cars statuses (in charge or not) in the preselected area, find the best station to suggest ensuring a uniform distribution.

3.3 Nonfunctional requirements

According to the needs of an average user, the system:

- Should be active 24/7
- Should be robust and reliable
- Should obey the law in terms of storing personal user’s data and credit card information
- Should provide an easy-to-use app with an attractive graphics

3.4 Scenarios

- Tim is a student visiting some relatives in the city. He is late for dinner and he needs to get from the hotel to his uncles’ home. Tim doesn’t want to grab a taxi because it will be too expensive for him, so he decides to use PowerEnJoy to drive to his uncles. After downloading and installing the app, he successfully registers and login. He immediately tries searching for a car near his hotel setting a distance of 200m. The app

notifies Tim that no car is found in the area around that input, so he extends the distance to 400m, but still he gets no result. Tim can finally find an available car at 800 meters away, so he makes a reservation.

- Adam is having a day off and wants to visit the suburbs of the city where there are a zoo and an aquarium. He knows public transports are poor in those areas so he books a car through PowerEnJoy and reaches it. Before unlocking the car, Adam, since he has read the instruction the first time he used the app, he checks the outside of the vehicle, and despite everything looked normal he notice a scratch on the top of the vehicle. Adam after getting inside the car and inserting correctly the PIN reports the scratch to the system indicating the location, just as suggested by the car display. Adam makes the ride to the zoo happy and confident that what he did will prevent him from some unexpected additional costs on the ride bill.

- Adam, after the zoo, uses the car of before to get to the aquarium. Since he will meet there with Kate, and she offered to take him home once the tour is finished, he's planning to leave the car just near the aquarium. When he's close to destination he starts "money saving" option, which shows him the nearest power station in the area. The car has 19% battery level, but Adam doesn't want to leave it at the station because it is too far for him. He then just parks in a safe area, that he is able to spot thanks to the clear distinction on the map, at 6 km away from the station. Adam is not surprised that he's been charged 30% more than the ride cost.

- John is an eco-caring person and for that reason has decided to reserve an electric car instead of using a combustion engine one. PowerEnJoy is exactly what he was looking for and because of that has chosen to keep the car for all the day-trip he and his girlfriend are doing in Milan. The day is now getting to the end and John has finished his trip so is looking for a place to park the car, so activates the "Saving mode" by setting his destination and following the instructions reaches a safe parking area provided with a recharge station, selects the "End ride and recharge car"

option reported on the car screen and in less than 1 minute plugs in the car putting it in charge. The car locks and John comes back home.

- Megan is a young model working for a well-known fashion brand. It's midday and her 2 hours' lunch break is starting so she decides to use the PowerEnJoy app to reserve a car and go to have lunch in a nice restaurant which is about 5 Km from her workplace. She finds a car with a charge rate of 40% but it's enough to her so she uses it. Arrived at the restaurant she finds a toll parking that the system lists as a safe area so she parks there and puts the car in a temporary park mode. Finished to eat, she comes back to the car, opens it and goes back to work.

- Lily is a law student that has to reach her friends to take an aperitif in Piazza Duomo. She reserved a car using the PowerEnJoy app half an hour ago and now she's reaching the car. She's looking in the nearby to find her car using a map provided by the system.
  Once found the car, she picks her smartphone out of her bag and uses the app to send an unlock command to the car. She presses the button "Unlock" and the car opens; now she can enter it and start her ride.

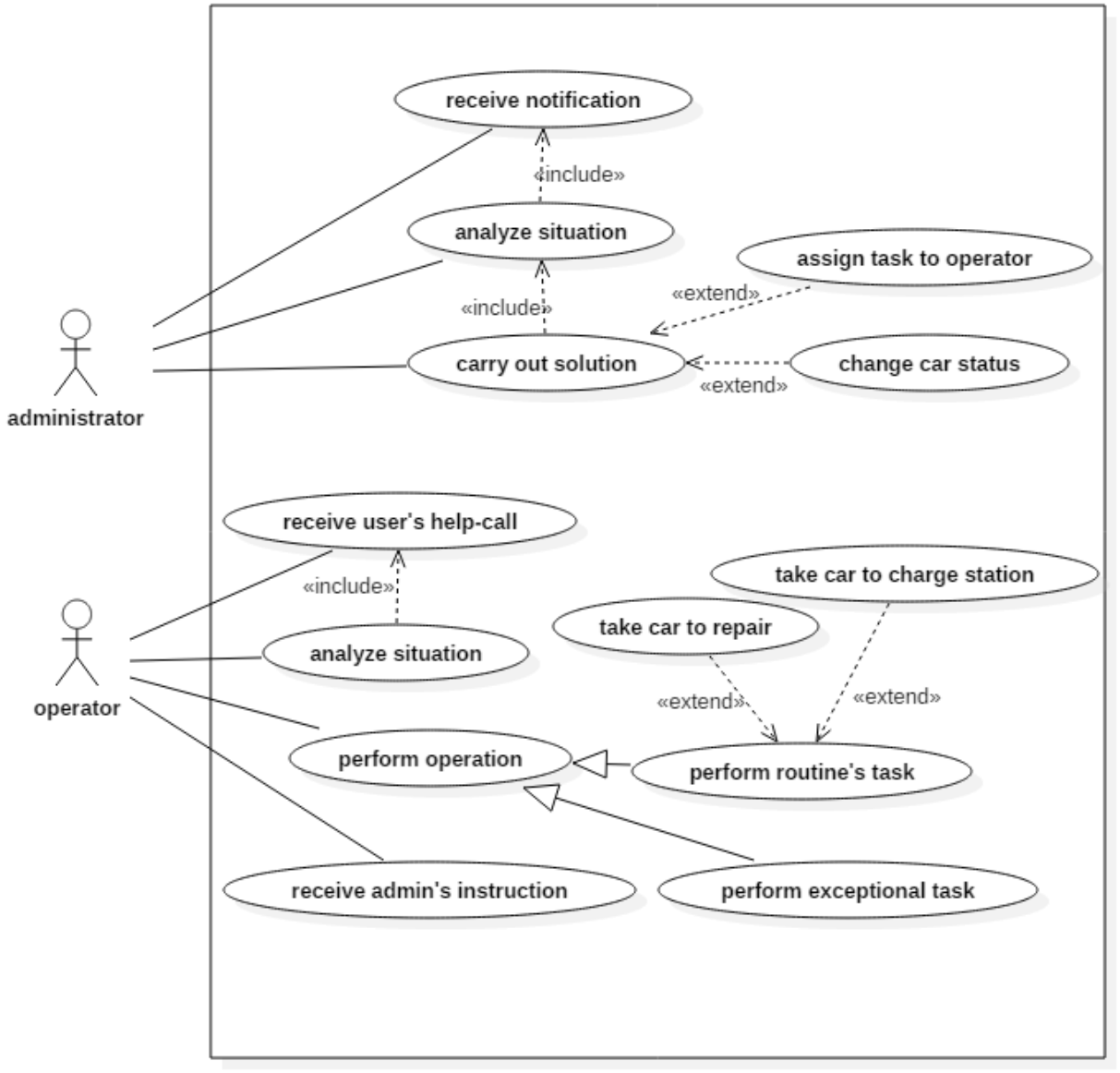
- Mark has just graduated at Politecnico di Milano and now wants to party with his friends. He has reserved a car and he's now having a ride in Milan.
  Given that two of his friends are not provided with a driving license he has to go and take them at their own houses. Mark reaches his first friend, Ted and picks him up, then goes and picks up even Barney, the second friend, and they all go and have fun together. At the end of the party they use the same car to come back home after having it temporary parked in front of a pub. Mark brings his friends home and then parks the car in a safe area. When he's leaving the car he notices that has received a discount for having brought with him two people. Being happy for having saved 2€ he decides to buy an ice cream.

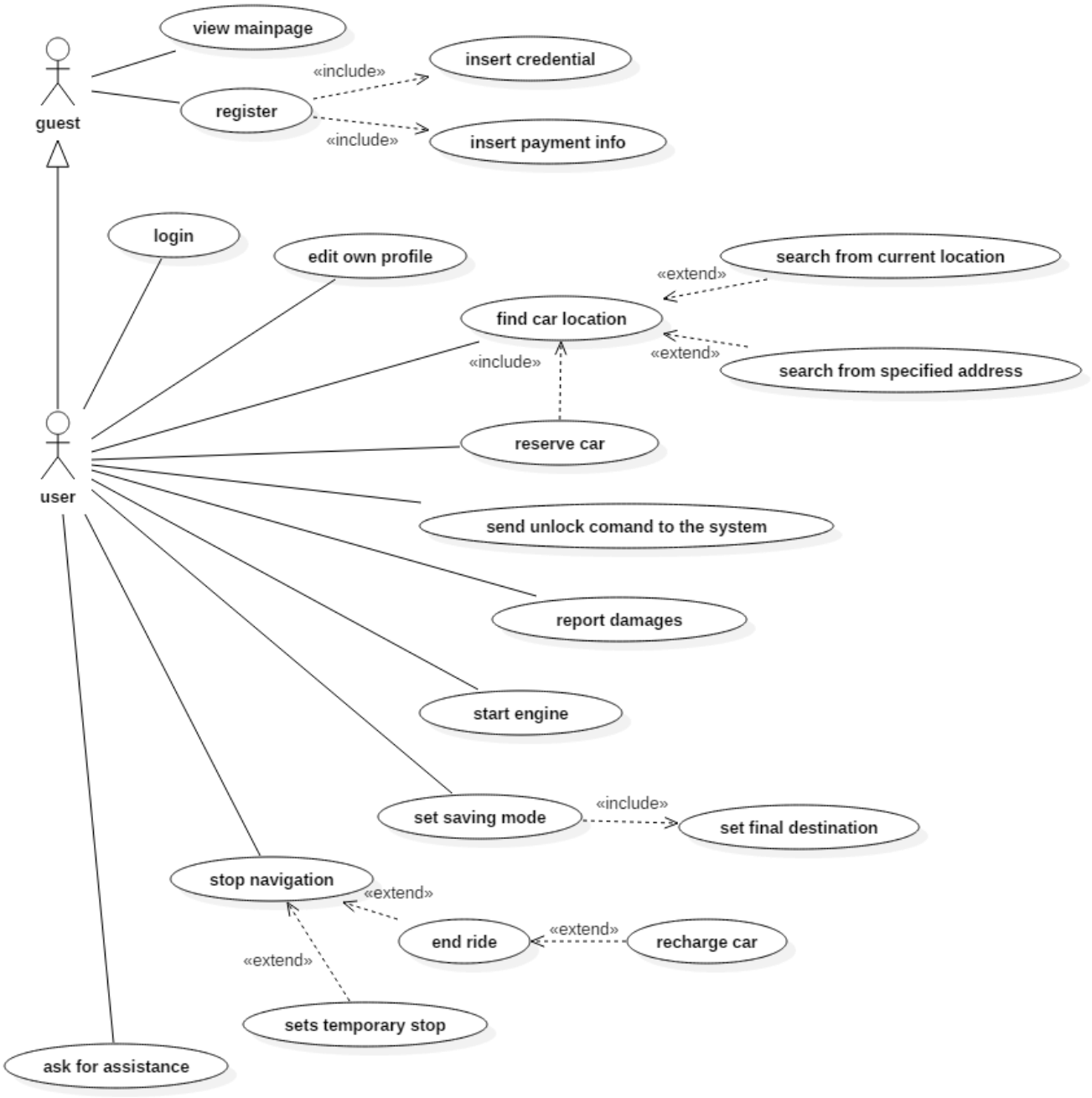
3.5 UML Models

Here below are provided the major UML diagrams that allow a concrete comprehension of how the system is going to operate.

3.5.1 Use case

Use cases involving user's and personnel activities are here reported. Given that the scope of the system is actually managing cars and users' actions, use cases related to administrators and operators are reported for completeness but not analyzed in detail.




guest
view mainpage
register
insert credential
insert payment info
«include»
«include»
user
login
edit own profile
search from current location
find car location
search from specified address
«extend»
«extend»
«include»
reserve car
send unlock comand to the system
report damages
start engine
set saving mode
set final destination
«include»
stop navigation
end ride
recharge car
«extend»
«extend»
«extend»
sets temporary stop
ask for assistance

3.5.1.1 View main page

Actors	Guest
Goal	Allow the visitor to collect information about PowerEnJoy and its service
Input condition	-
Event flow	1. The guest accesses PowerEnJoy web site 2. The guest visits web pages and collects information about the service 3. The guest decides whether to register or leave the web site
Output condition	-
Exceptions	-

3.5.1.2 Register

Actors	Guest
Goal	Allow the visitor to register to PowerEnJoy by submitting a form containing his/her personal data and payment information
Input condition	-
Event flow	1. The guest clicks "Register" in the web page 2. The guest fills and submits his/her personal data and credit card number to the system 3. The system verifies user's data and releases a password to access PowerEnJoy infrastructures
Output condition	Guest is now a user and the system has provided him with a password to access the service
Exceptions	• User data are not valid • User credit card is not valid

3.5.1.3 Login

Actors	User
Goal	Allow the user to fill his credentials and access the system
Input condition	-
Event flow	1. The user clicks "Login" in the web page 2. The user fills in his/her credentials 3. The user presses the "Login" button
Output condition	The user is logged in
Exceptions	• Username is invalid • Password is invalid

3.5.1.4 Edit own profile

Actors	User
Goal	Allow the user to edit his profile data.
Input condition	The user is logged in
Event flow	1. The user choses "Edit profile" from his/her account options 2. The user modifies his/her personal data and presses "Confirm" 3. New data are processed by the system and, if correct, accepted
Output condition	User's data have been modified
Exceptions	• Incorrect data inserted

3.5.1.5 Find car location

Actors	User
Goal	Allow the user to find a car location
Input condition	The user is logged in
Event flow	1. The user accesses the PowerEnJoy app 2. The user selects "Find a car" among all possible actions 3. The user submits the system his/her location. Location may be specified by using user's current position or by entering a valid address 4. The system looks for cars in the nearby of the specified positions having the status set as "AVAILABLE" or "RECHARGING" and a battery level of at least 15%.
Output condition	User is provided with a map showing every usable car in the nearby of the specified position. Every found car is ensured to be available.
Exceptions	• Location entered doesn't exist • Impossible to detect user's position

3.5.1.6 Reserve car

Actors	User
Goal	Allow the user to reserve a car for 1h time
Input condition	The user is logged in and has already queried the system to find a car
Event flow	1. The user selects one of the cars proposed by the system 2. The user presses the button "Reserve" 3. The car status is changed to "BOOKED"
Output condition	The selected car is reserved for up to 1h
Exceptions	• The user is trying to reserve more cars for the same period

3.5.1.7 Send unlock command to the system

Actors	User
Goal	Allow the user to unlock a reserved car
Input condition	The car has been reserved and the user took less than 1h to use it
Event flow	1. The user reaches the car 2. The user unlocks the car using PowerEnJoy by pressing the button "Unlock car" 3. The user has access to the car up to 1 minute 4. The system starts charging money to the user
Output condition	The car is unlocked
Exceptions	• The 1 minute after unlocking expires and the user has not entered the car yet. - The system locks again the car.

3.5.1.8 Report damages

Actors	User
Goal	Allow the user to report possible damages related to the car before starting using it
Input condition	The car has been unlocked and the user is inside it
Event flow	1. Ask the user to report the car status by expressing a vote (1 to 5 stars) 2. Record the vote expressed and eventually collect further information
Output condition	The ride is ready to start
Exceptions	-

3.5.1.9 Set saving mode

Actors	User
Goal	Allow the user to get tips for saving money by applying virtuous behaviors
Input condition	The car is "ONROAD"
Event flow	1. The user presses the button "Saving mode" 2. The system tells the user which could be useful tips to save money at the end of the ride. The user is not bound to that tips and doesn't necessarily follow them. Tips are signaled taking into account user's destination. If no destination is inserted into the car system, the user is asked to provide one.
Output condition	-
Exceptions	-

3.5.1.10 Start engine

Actors	User
Goal	Allow to user to start the engine and begin his/her ride
Input condition	The car is unlocked
Event flow	1. The user turns the key of the car 2. The system changes the car status from "PARKING" to "ONROAD"
Output condition	The car is unlocked and the ride has started
Exceptions	• Mechanical fault

3.5.1.11 Stop navigation

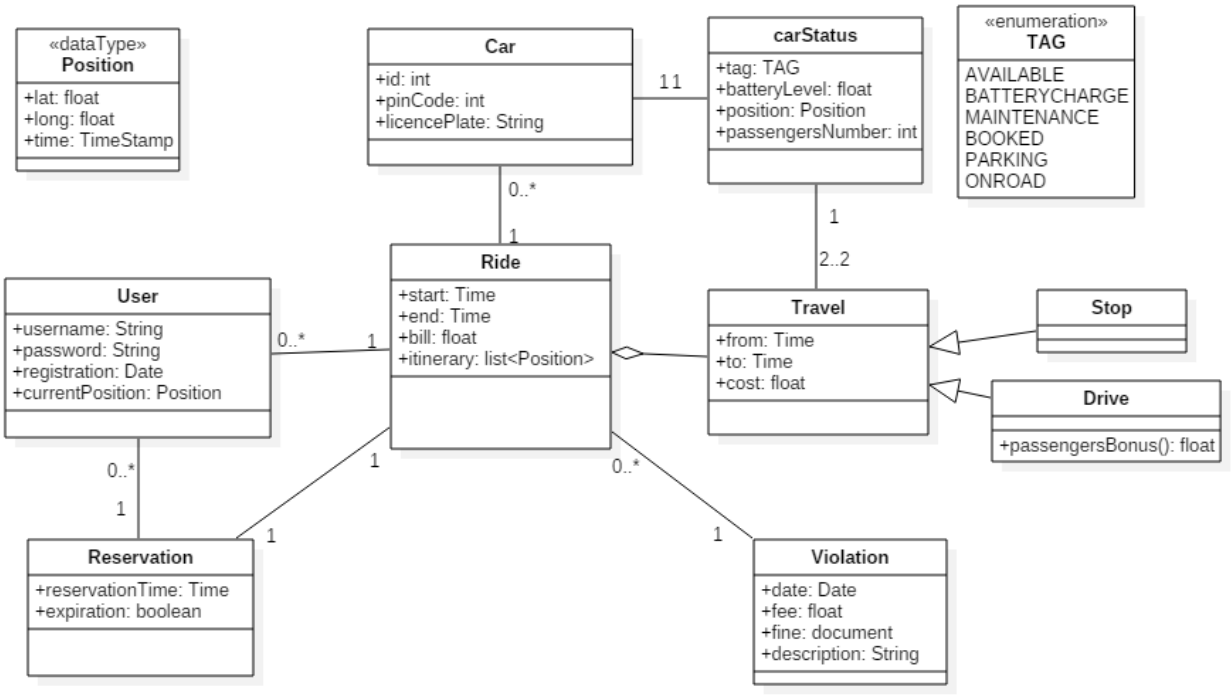
Actors	User
Goal	Allow to user to stop his/her navigation for a temporary of definitive stop
Input condition	The car is "ONROAD"
Event flow	1. The user chooses whether to park the car or to end his/her ride by clicking the appropriate button shown on the car screen 2. **A)** If the user chooses to do a temporary stop, the car status is changed to "PARKING" **B)** If the user chooses to end the ride, the status is changed to "AVAILABLE" **C)** If the user chooses to end the ride and recharge the car (s)he has 1 minute, after the engine is turned off, to plug it in a recharge station. When a car is recharging the status is set to "BATTERYCHARGE". If the user takes more than 1 minute to plug in the car, the recharge option is discarded and the status is set to "AVAILABLE" 3. In cases **B** and **C**, a payment request is submitted to the system 4. The car is locked by the system as soon as the user: - gets off (cases 2 and 3) - takes more than 1 minute to plug in the car in the recharge station - plugs in the car in the recharge station
Output condition	The car is parked, recharging or available
Exceptions	-

3.5.1.12 Ask for assistance

Actors	User
Goal	Allow the user to receive assistance in case of problems or technical issues
Input condition	-
Event flow	1. The user presses the button "Ask for assistance" on the car screen 2. The user is show some FAQ 3. If no FAQ is useful, the user can be put in contact with a PowerEnJoy operator
Output condition	-
Exceptions	-

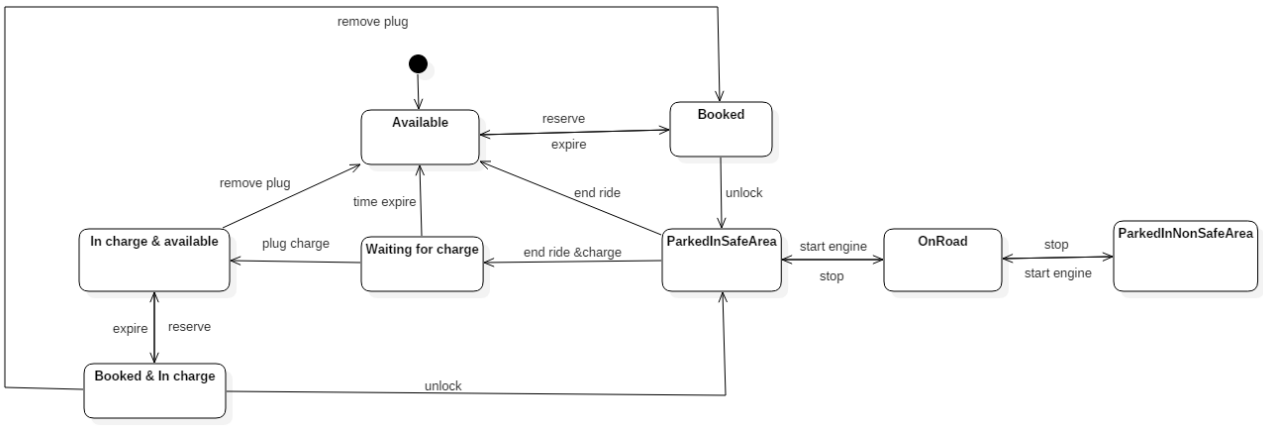
3.5.2 Class diagram

Here below is reported a complete schema concerning how the system has been thought.



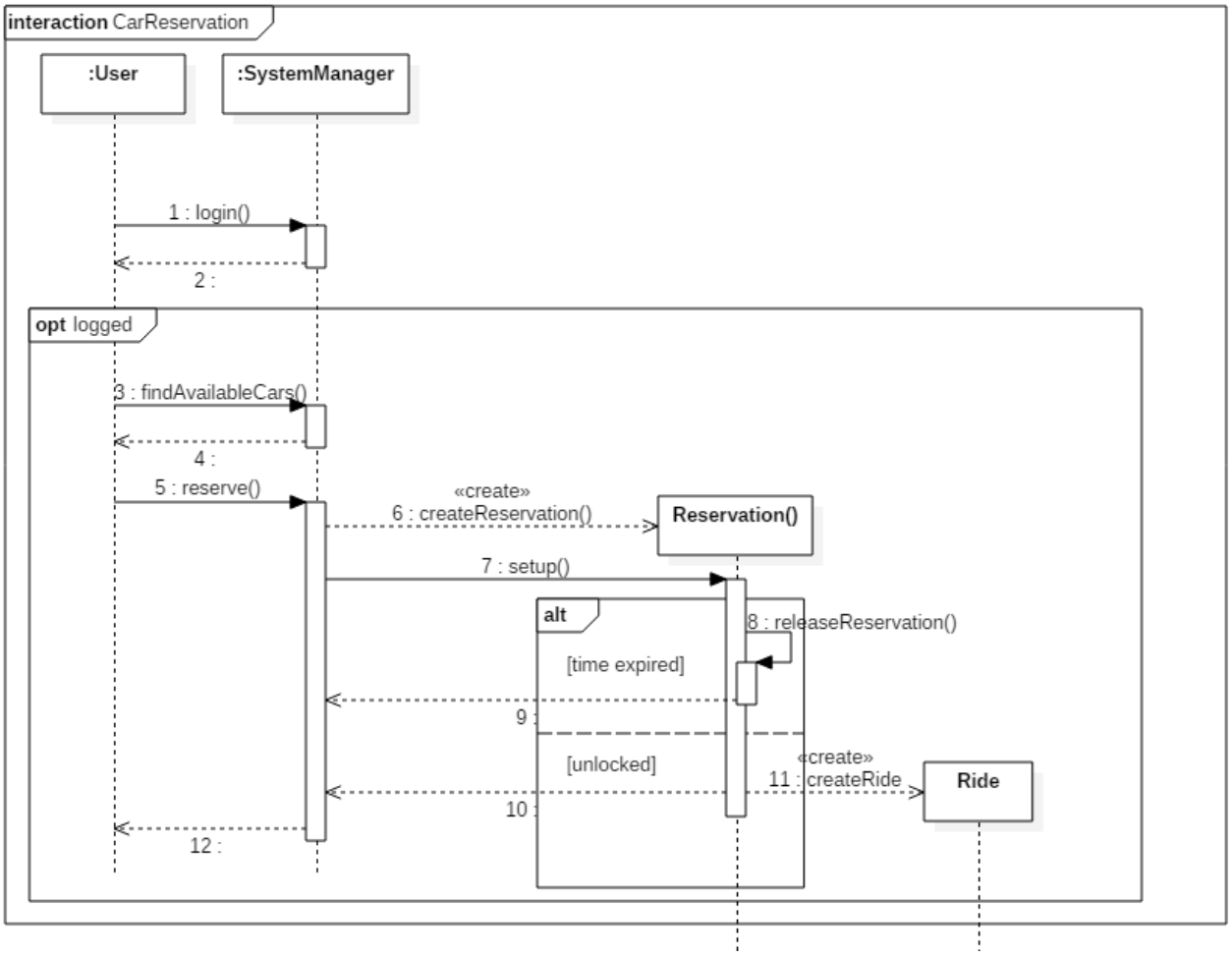
3.5.3 State diagrams

Here is represented a state diagram concerning the car activity and expressing every possible state in which a car can be found.



3.5.4 Sequence diagrams

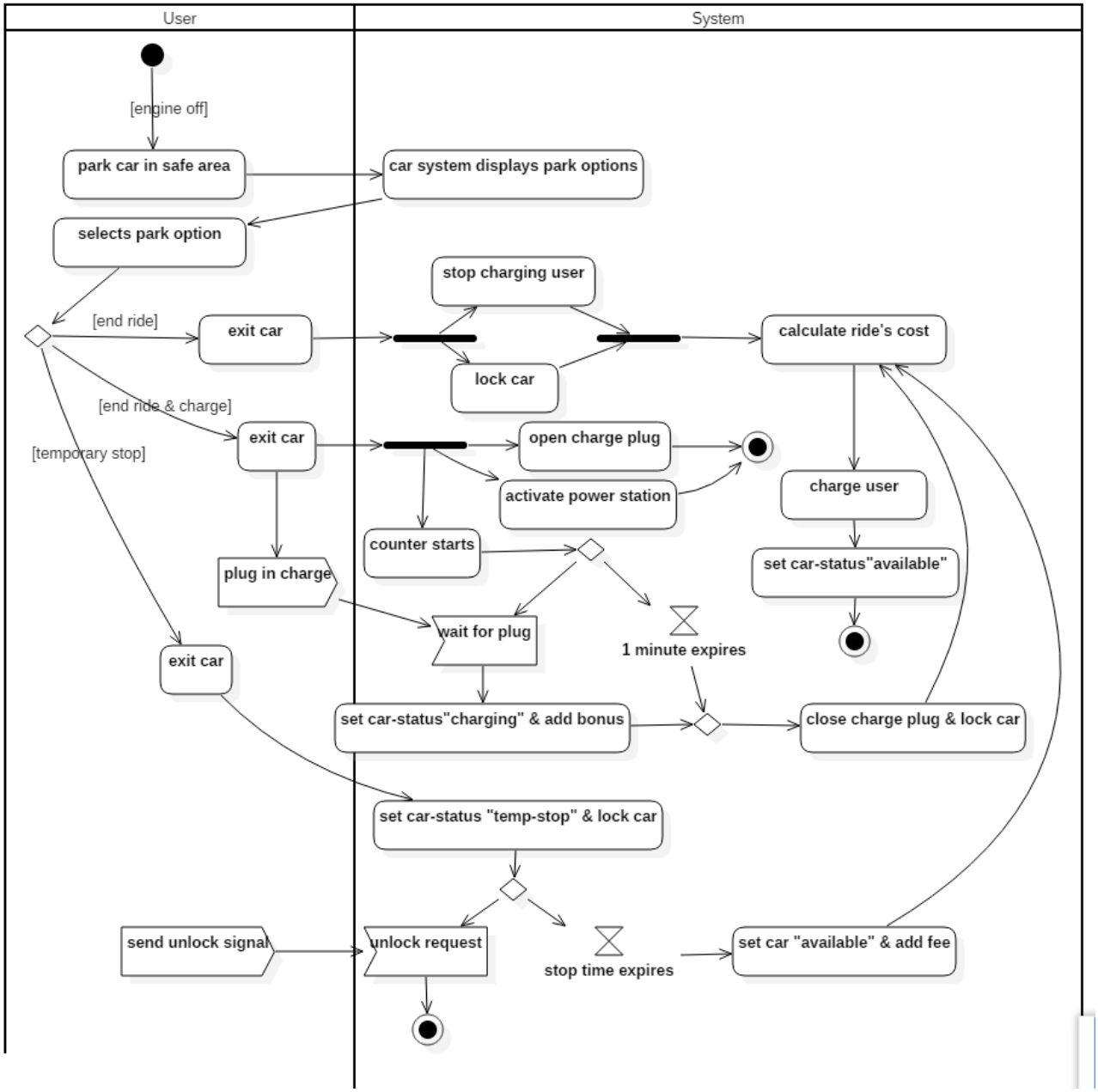
Here is presented a sequence diagram expressing the how the procedure for unlocking a car has been thought.



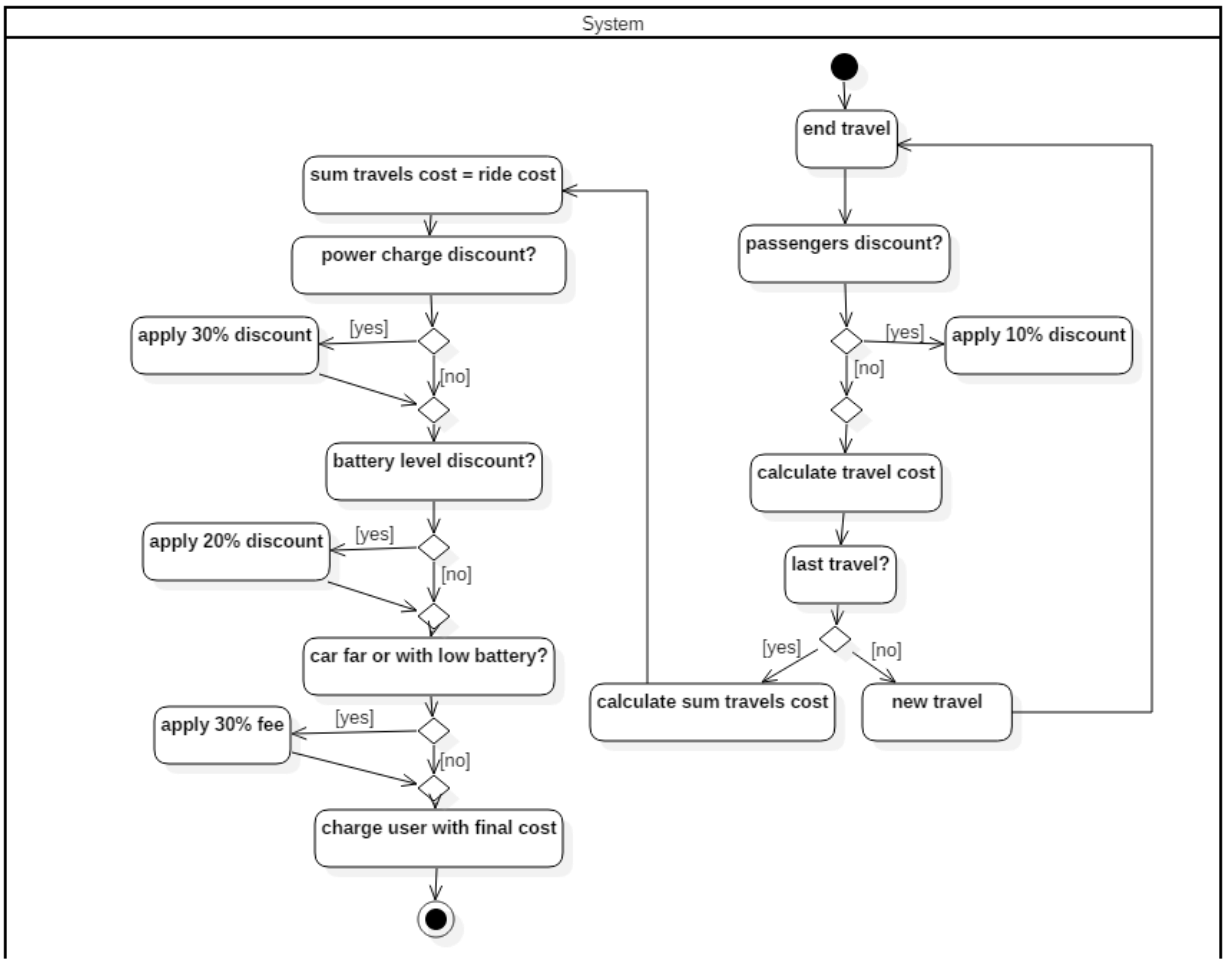
3.5.5 Activity diagrams

Here are reported three activity diagrams involving respectively:

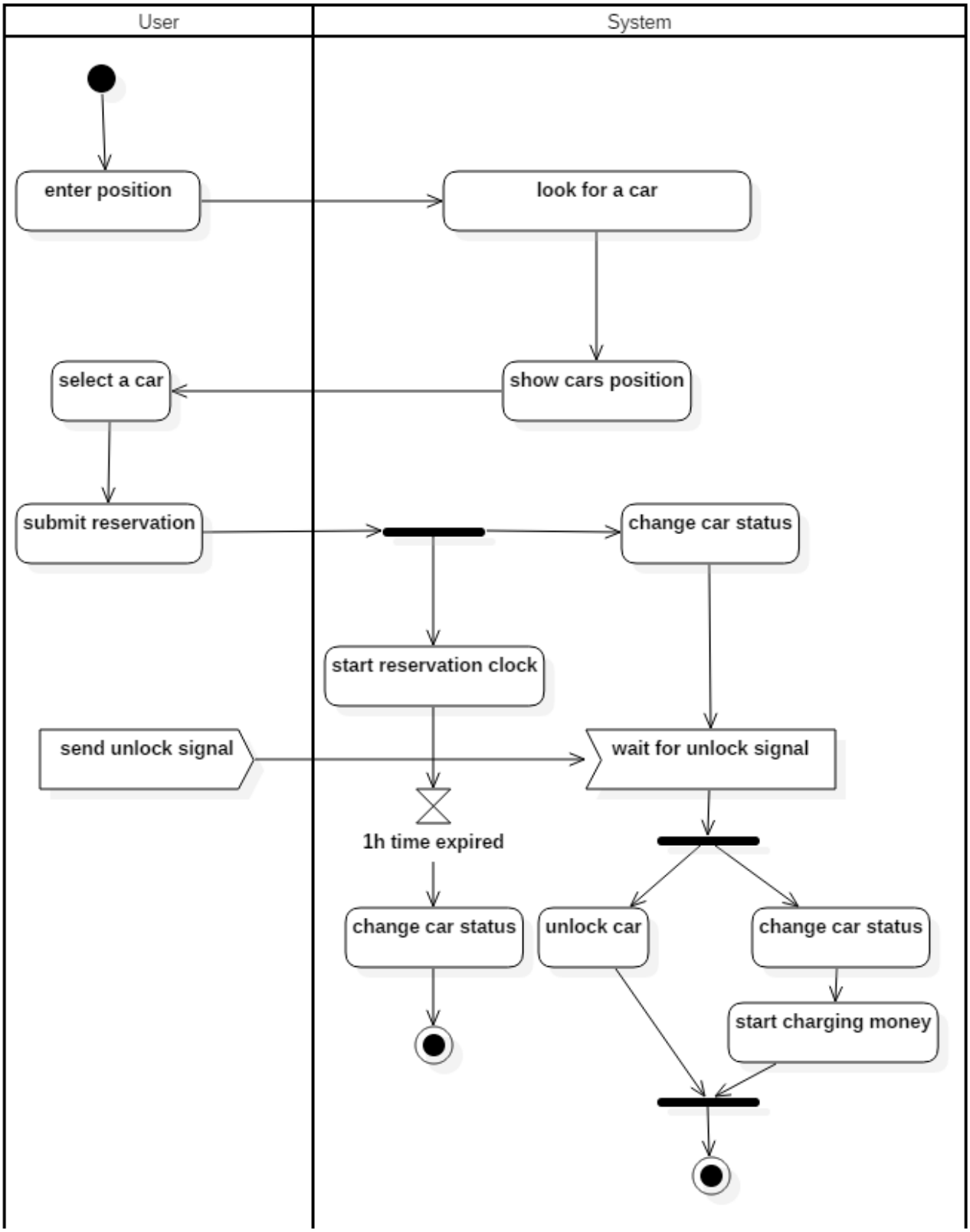
1) Parking activity



2) Cost calculation



3) Car reservation and unlocking



4 Alloy modeling

4.1 Alloy

Here has been reported the result obtained using Alloy as modeling to prove the consistency of the proposed model.

4.1.1 Signatures

```
sig User {}

sig Car{
	status: one CarStatus
}

sig CarStatus{
	tag: one Tag,
	batteryLow: lone Bool,
	distanceGreater: lone Bool,
	passengerNumber: one Int,
}{
	(tag = ONROAD) => (passengerNumber > 0 && passengerNumber <= 4)
	(tag != ONROAD) => (passengerNumber = 0)
	(distanceGreater != none ) <=> (tag = AVAILABLE)
	(batteryLow != none ) <=> (tag = AVAILABLE)
}

abstract sig Tag {}
one sig AVAILABLE extends Tag {}
one sig BATTERYCHARGE extends Tag {}
one sig BOOKED extends Tag {}
one sig PARKING extends Tag {}
one sig ONROAD extends Tag {}


sig Reservation{
	user: one User,
	car: one Car,
	expired: one Bool,
	ride: lone Ride
}{
	(ride != none) <=> (expired = False)
	(expired=True)  => (car.status.tag=AVAILABLE || car.status.tag=BATTERYCHARGE)
}
```

```
sig Ride{
        user: one User,
        car: one Car,
        stop: some TravelStop,
        drive: some TravelDrive,
        finalStatus: lone CarStatus,
        violation:set Violation,
        chargeBonus: one Bool,
        batteryBonus: one Bool,
        negativeBonus: one Bool
}{
        finalStatus!=none<=>(finalStatus.tag=AVAILABLE||finalStatus.tag=BATTERYCHARGE)
        chargeBonus=True <=>finalStatus.tag=BATTERYCHARGE
        batteryBonus=True <=> (finalStatus.tag=AVAILABLE && finalStatus.batteryLow=False)
        negativeBonus=True <=>(finalStatus.distanceGreater=True || finalStatus.batteryLow=True)
}


abstract sig Travel{
        carStatus : some CarStatus
}{
        #carStatus=2
}

sig TravelStop extends Travel{
}{
        carStatus.tag = PARKING
}

sig TravelDrive extends Travel{
        passengerBonus: one Bool
}{
        carStatus.tag = ONROAD
        (passengerBonus=True) <=> (carStatus.passengerNumber=3 || carStatus.passengerNumber=4)
}

sig Violation{}
```

4.1.2 Facts

Each ride has got only one final state

```
fact oneFinalStatusForRide{
        no r1,r2:Ride | r1!=r2 && r1.finalStatus=r2.finalStatus
}
```

Users and cars are unique in reservation

```
fact noMoreUserAndCarSameReservation{
        no r1,r2:Reservation| (r1!=r2) && (r1.user=r2.user)
        no r1,r2:Reservation|(r1!=r2) && (r1.car=r2.car)
 }
```

Users and cars are unique in reservation

```
fact noMoreUserAndCarSameRide{
        no r1,r2:Ride | (r1!=r2) && (r1.user=r2.user)
        no r1,r2:Ride | (r1!=r2) && (r1.car=r2.car)
}
```

Only one ride is related to a not expired reservation

```
fact noRideWithoutReservation{
        no res1,res2:Reservation | res1 != res2 && res1.ride=res2.ride
        all ride:Ride | userReservation[ride.user]
        all r:Ride,res:Reservation | (r.user=res.user) => (r.car=res.car)
        no r:Ride,res:Reservation | res.ride = r && res.user != r.user
}

pred userReservation[user1:User]{
        one reservation:Reservation | reservation.user=user1 && reservation.expired=False
}
```

Violation is unique

```
fact noMoreRideSameViolation{
        no disjoint r1,r2:Ride | r1.violation & r2.violation !=none
}
```

Travels are unique

```
fact noMoreRideSameTravel{
        no disjoint r1,r2:Ride | r1.stop & r2.stop != none
        no disjoint r1,r2:Ride | r1.drive & r2.drive != none
        all travelStop:TravelStop | notAloneStop[travelStop]
        all travelDrive:TravelDrive | notAloneDrive[travelDrive]
}

pred notAloneStop[travelStop:TravelStop]{
        one ride:Ride | ride.stop&travelStop!=none
}


pred notAloneDrive[travelDrive:TravelDrive]{
        one ride:Ride | ride.drive&travelDrive!=none
}
```

Car status is unique both in car and in travel

```
fact noMoreCarSameStatus{
        no car1,car2:Car | car1!=car2  && car1.status=car2.status
        all carStatus:CarStatus | notAloneStatus[carStatus]
        all car:Car | statusUnique[car]
}

pred notAloneStatus[carStatus1:CarStatus ]{
        (one travel:Travel | travel.carStatus&carStatus1!=none) ||
        (one car:Car | car.status&carStatus1!=none)
}

pred statusUnique[car1:Car]{
        no ride:Ride | ride.car!=car1 && (car1.status&ride.stop.carStatus!=none)
        no ride:Ride | ride.car!=car1 && (car1.status&ride.drive.carStatus!=none)
        no ride:Ride | ride.car!=car1 && (car1.status&ride.finalStatus!=none)
}
```

```
fact noMoreTravelSameStatusAndDifferentCar{
        no disjoint ts1,ts2: TravelStop | ts1!=ts2 && ts1.carStatus & ts2.carStatus !=none
        no disjoint td1,td2: TravelDrive | td1!=td2 && td1.carStatus & td2.carStatus !=none
        no disjoint td1: TravelDrive,ts2: TravelStop |   td1.carStatus & ts2.carStatus !=none
}
```

The cars with (onRoad-Parking-Booked) status belong to one reservation

```
fact noCarsOnRoadNotBelongingToReservationOrRide{
        no c: Car  | noReservationOrRide[c]  && c.status.tag = ONROAD
        no c: Car  | noReservationOrRide[c]  &&  c.status.tag = PARKING
        no c: Car  | noReservationOrRide[c]  &&  c.status.tag = BOOKED
}

pred noReservationOrRide[c: Car]{
        no res: Reservation, ride: Ride | res.car = c || ride.car = c
}
```

4.1.3 Assertions

All bonus are assigned as defined in the requirements's specification

```
assert bonusMeansEndRide{
        no r: Ride | r.batteryBonus = True && r.finalStatus = none
        no r: Ride | r.chargeBonus = True && r.finalStatus = none
        no r: Ride | r.negativeBonus = True && r.finalStatus = none

        all r: Ride | (r.chargeBonus = True) implies (r.car.status.tag = BATTERYCHARGE)
        all r: Ride | (r.batteryBonus = True) implies (r.car.status.tag = AVAILABLE &&
                                                        r.car.status.batteryLow = False)
        all r: Ride | (r.negativeBonus = True) implies (r.car.status.batteryLow = True||
                                                        (r.car.status.distanceGreater = True))
}

check bonusMeansEndRide
```

the expiration process cannot be stopped and his effect is unchangeable

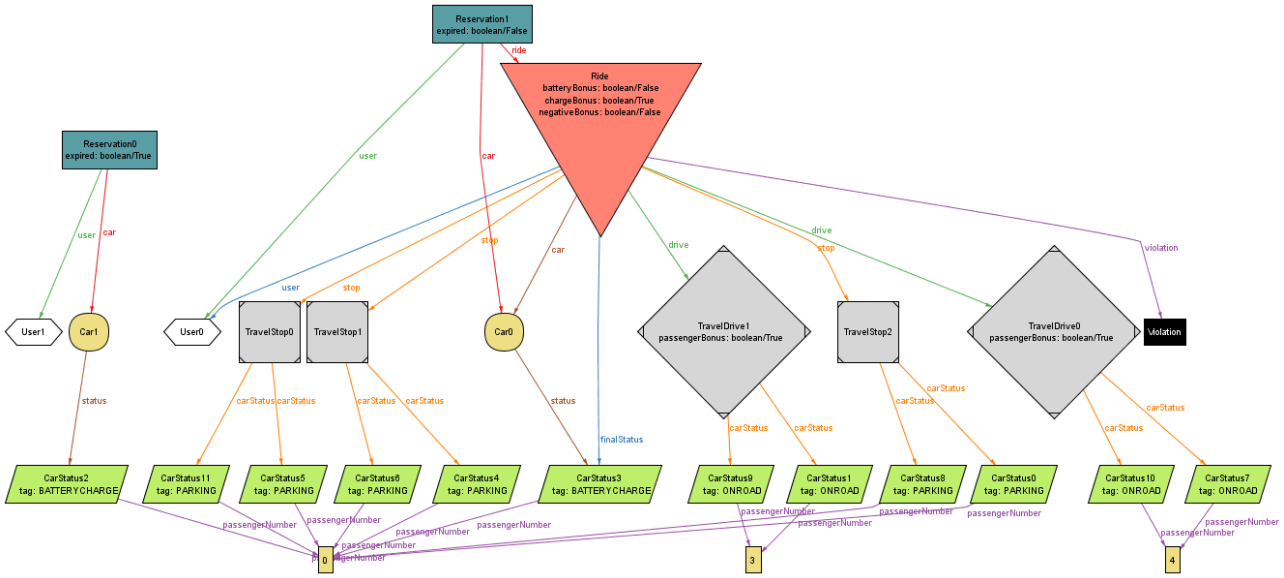
```
assert expiredReservationImpliesNoRide{
        all res: Reservation | (res.expired = True) implies (noRideRelated[res])
}

pred noRideRelated[res: Reservation]{
        no r: Ride | r = res.ride
}

check expiredReservationImpliesNoRide
```

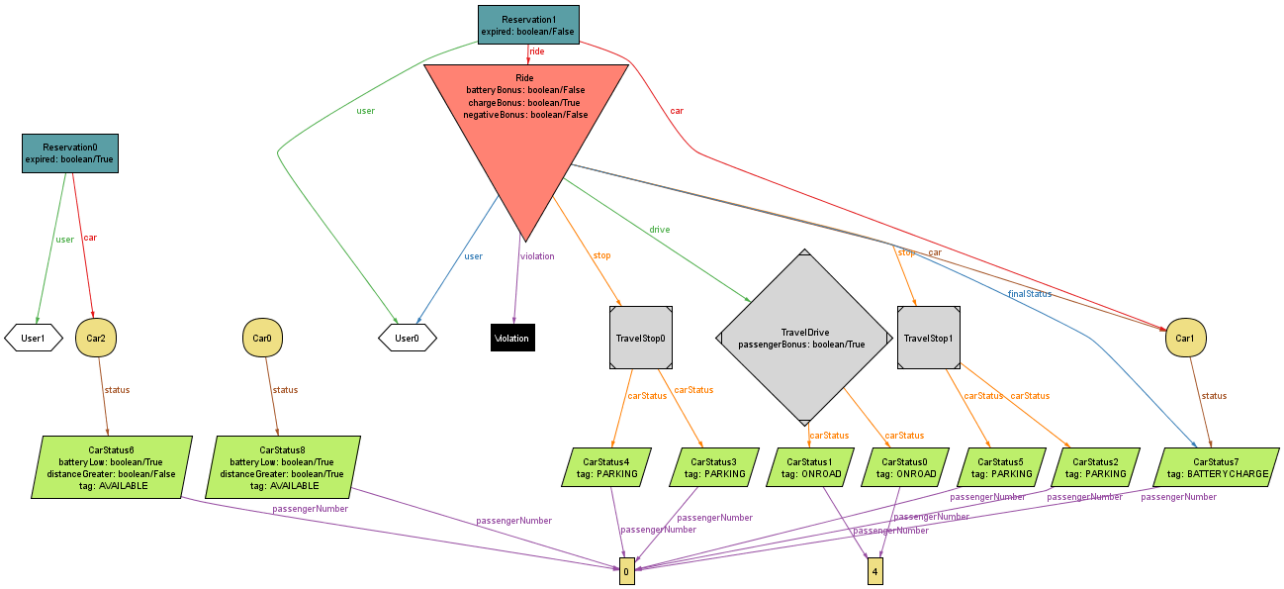
Executing "Check bonusMeansEndRide"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
4037 vars. 228 primary vars. 8505 clauses. 151ms.
No counterexample found. Assertion may be valid. 30ms.

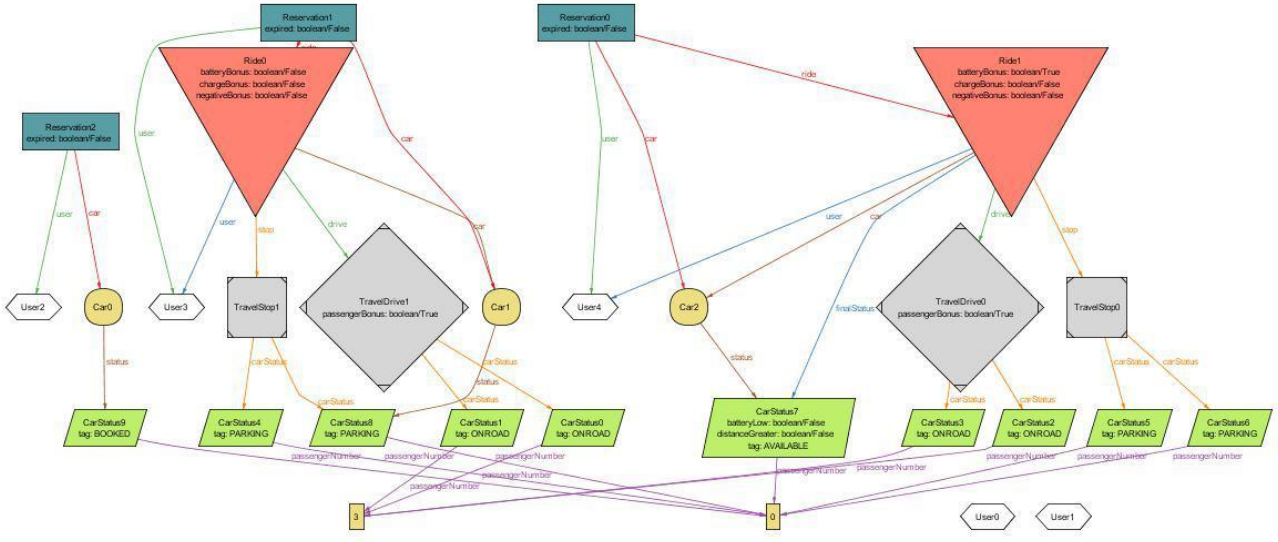
Executing "Check expiredReservationImpliesNoRide"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3870 vars. 234 primary vars. 8111 clauses. 144ms.
No counterexample found. Assertion may be valid. 59ms.

Executing "Check rideMeansNoExpiredReservation"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3855 vars. 231 primary vars. 8123 clauses. 100ms.
No counterexample found. Assertion may be valid. 44ms.

4.1.4 Generated world




Reservation1
expired: boolean/False
ride
Ride
batteryBonus: boolean/False
chargeBonus: boolean/True
negativeBonus: boolean/False
user
car
Reservation0
expired: boolean/True
user
car
violation
stop
drive
stop
car
finalStatus
User1
Car2
Car0
User0
Violation
TravelStop0
TravelDrive
passengerBonus: boolean/True
TravelStop1
Car1
status
status
status
carStatus
carStatus
carStatus
carStatus
carStatus
carStatus
CarStatus6
batteryLow: boolean/True
distanceGreater: boolean/False
tag: AVAILABLE
CarStatus8
batteryLow: boolean/True
distanceGreater: boolean/True
tag: AVAILABLE
CarStatus4
tag: PARKING
CarStatus3
tag: PARKING
CarStatus1
tag: ONROAD
CarStatus0
tag: ONROAD
CarStatus5
tag: PARKING
CarStatus2
tag: PARKING
CarStatus7
tag: BATTERYCHARGE
passengerNumber
0
4


Reservation1
expired: boolean/False
Ride0
batteryBonus: boolean/False
chargeBonus: boolean/False
negativeBonus: boolean/False
Reservation0
expired: boolean/False
ride
Ride1
batteryBonus: boolean/True
chargeBonus: boolean/False
negativeBonus: boolean/False
Reservation2
expired: boolean/False
user
car
stop
drive
finalStatus
User2
Car0
User3
TravelStop1
TravelDrive1
passengerBonus: boolean/True
Car1
User4
Car2
TravelDrive0
passengerBonus: boolean/True
TravelStop0
status
carStatus
CarStatus9
tag: BOOKED
CarStatus4
tag: PARKING
CarStatus8
tag: PARKING
CarStatus1
tag: ONROAD
CarStatus0
tag: ONROAD
CarStatus7
batteryLow: boolean/False
distanceGreater: boolean/False
tag: AVAILABLE
CarStatus3
tag: ONROAD
CarStatus2
tag: ONROAD
CarStatus5
tag: PARKING
CarStatus6
tag: PARKING
passengerNumber
3
0
User0
User1

5 Revision

5.1.1 Software and tools used

The following software have been used:

- Alloy (Alloy modeling)
- Balsamiq (mockup design for interfaces)
- Microsoft Word (document writing)
- Star UML (UML diagrams)

5.1.2 Team work

Here is reported a compact table showing how the work was brought on by all the members of the group.

Member	*Hours of work*
Cattaneo Davide	35
El Hariry Matter	44
Frontino Francesco	43