Politecnico di Milano

A.A. 2016-2017

Software Engineering 2: PowerEnJoy

Requirement **A**nalysis and **S**pecification **D**ocument

Lorenzo Frigerio (mat. 879203)

Martina Perfetto (mat. 808869)

Ivan Vigorito (mat. 879204)

3rd February 2016

Version 1.3

Contents

1. Introduction 4
 1.1. Purpose 4
 1.2. Scope 4
 1.3. Glossary: 5
  1.3.1. Acronyms 7
 1.4. Goals 7
 1.5. Assumptions 8
  1.5.1. General Assumptions 8
  1.5.2. Invoicing Assumptions 9
 1.6. Constraints 9
 1.7. Reference documents 10
 2. Specific Requirements 11
 2.1. Functional Requirements 11
  2.1.1. [G1] Users must be able to register to the system by providing their credentials and payment information 11
  2.1.2. [G2] Registered users must be able to find the locations of available cars within a certain distance from their current location or from a specified address 11
  2.1.3. [G3] Users must be able to reserve cars 12
  2.1.4. [G4] A user must be able to enter in the car reserved by him when nearby 12
  2.1.5. [G5] The user is able to know the current charge of the ride 12
  2.1.6. [G6] The system is able to calculate the final cost of the rental 13
  2.1.7. [G7] The system is able to know when a ride ends 13
  2.1.8. [G8] The system is able to manage the cars 13
  2.1.9. [G9] The user is able to update his profile and payment methods 13
 2.2. Non-Functional Requirements 14
  2.2.1 User Interfaces: Login 14
  2.2.2 User Interfaces: Registration 15
  2.2.3. User Interfaces: Search & Reserve 16
  2.2.4 User Interfaces: map and cars 17
  2.2.5. User Interfaces: mobile application login and settings 18
  2.2.6. User Interfaces: mobile application, rental stage 19

2.2.7. User Interfaces: mobile application, end of rental 20
2.2.8. User Interfaces: mobile application, unlock the car 21
2.2.9. Usability 21
2.2.10. Privacy requirements 22
2.3. The world and the Machine 22
3. Software Design – UML 23
3.1. Actors definition 23
3.2. Possible scenarios 23
3.3. Actor: Unregistered user 25
3.4. Actor: Car 27
3.5. Actor: registered user 32
3.6. Actor: Employee 42
3.7. Domain Model 45
3.7.1. Domain Class Diagram 45
3.7.2. Main dynamics of the system 47
3.7.3. Main sequences of the system 49
3.7.4. State Chart Diagram 52
4. Model Coherence Analysis 54
4.1. Alloy Code 54
4.2. Generated World 58
5. Tools 61
5.1. Hours of work 62
5.2. ChangeLog 62

1. Introduction

1.1. Purpose

The purpose of the system is to offer a car sharing service to users that want to move around a city. The system includes the use of only electric cars in order to promote sustainable mobility inside cities taking advantage of the continuously increase in the number of power grid stations. Moreover, it will stimulate the buying of electric cars due to the positive experience users will have during their journeys. Target users are people over 18 years old who have a driving licence and particularly young people that are accustomed to these new type of services and mobile applications. The system will extend already present similar services but with the peculiarity of electric cars and a smart orientation that favours sharing cars among multiple users and parking in special areas with significant discounts.

1.2. Scope

The aim of the project is to create a new online car-sharing system using only electric cars. Users register themselves online providing their credentials (name, surname, email and others) and selecting a method of payment. After that they receive back a personal password that can be used to access to the system together with the associated email. Logged user is able to look for cars either around his position or a certain address inserted, selecting from different ranges of distance. Then he can choose and reserve one car from the list proposed by the system, which must be reached within an hour. If the user does not unlock the car in one hour from the reservation, he will be charged a fee of 1 EUR. When the user is inside a radius of 20 meters from the reserved car he could ask to the system to unlock the car. Then the system will check the distance between the car and the user, who must be the owner of the reservation.

As soon as the engines ignites due to a push of the start/stop button, the system starts charging the user for a fee of 0,25 EUR/min

All the electric cars are equipped with a GPS navigation device, where appears the percentage of the battery and the updated amount he is paying.

Whenever the user leaves the car and no passengers are inside, the car is automatically closed. In addition, through the application after exiting from the car, the user can select to terminate the ride and a recap of the total costs will appear on the application. In this case the car will be set available again by the system.

Users can park anywhere inside the safe areas, which may contain a special parking area with a power grid station.

A map with all the safe areas is available on the website.

In order to facilitate the parking, safe areas are whole cities (such as the metropolitan area of Milan), so users spend less time to find a suitable place.

1.3. Glossary:

User / registered user: he/she is the client of the service; he/she is able to rent a car in order to travel around the city. He is associated with:

-Name

-Surname

-Other personal information

-Method of payment

-Number of driving licence and expiration date

-Password

Employee/operator: is the one that help users in case of emergency and has the responsibility of managing cars in case of malfunction. Users can call them by using the telephone exchange of the application.

Method of payment: is inserted by the user during the registration phase but can be updated over the time. Only one method is active at once and payment are concluded using services offered by the different companies holding the credit card. An invoice containing all the charges collected is generated monthly.

Car: sometimes referred as vehicle is the means of transport rented by users. It contains a set of sensors that analyse the number of passengers presents on the car, control the charge of the battery and detect when a door is closed. Moreover, it includes a module that transmit this information to the system using the Internet.

Available car: car that is not in use at the moment by any user, has at least 20% of charge and is not reserved by anyone.

Reservation: made by a user that wants to use a car. Has a duration of one hour maximum and is associated with a unique car. Once the user asks to unlock the car the car becomes associated to the user until he decides to end the ride.

Charge: amount of money that users have to pay due to the use of the service. It is immediately calculated by the system after a ride but money is transferred only at the end of the month.

Penalty: fee derived from a bad behaving of the user such as a damage on the car or a fine for exceeding speed limits. The fee will be notified to the user and included in the monthly invoice.

GPS navigation device: system that equip each car and that is able to calculate the exact position of the car and display to the user the route to follow. Its display is also used to show the current fee of the ride and the status of the battery.

Start/stop button: device present in all the electric cars that allow the engine to ignite when the car is unlocked. It also allows the engine to stop when the user wants to get off the car.

Special Safe Area: special parking areas that contain plugs that allow cars to be recharged. They are provided with sensors that detect the number of spots that are currently used and communicate the number to the system. They are also called power grid stations.

Safe Area: Space included in boundaries that determine where users can park a car. It covers entire metropolitan cities in order to facilitate users to find a park and they may also contain power grid stations. Users cannot terminate a ride while outside from a safe area.

Ride/Rental: it last from when the user picks up the car until when the system stops charging the user. It includes a possible set of temporary stops and the total path travelled by the user.

Distance: It is the lengthiness of road which is covered by the user with the rented car.

Park: is when a user leaves the car and wants to end the rental. At this point the system stops charging the user.

Stop: is when a user leaves the car but wants to resume the ride in the future. The car will be locked by the system that, however, will continue to charge the user for the ride.

1.3.1. Acronyms

[Gn]: n-goal.

[Rn]: n-functional requirement.

[Dn]: n-domain assumption.

SPA: Special Safe Areas

SA: Safe Areas

1.4. Goals

The main goals identified and that the system should be able to provide are:

[G1] Users must be able to register to the system by providing their credentials and payment information.

[G2] Registered users must be able to find the locations of available cars within a certain distance from their current location or from a specified address.

[G3] Users must be able to reserve cars.

[G4] A user must be able to enter in the car reserved by him when nearby.

[G5] The user is able to know the current charge of the ride.

[G6] The system is able to calculate the final cost of the rental.

[G7] The system is able to know when a ride ends.

[G8] The system is able to manage the cars.

[G9] The user is able to update his profile and payment methods.

1.5. Assumptions

In this paragraph, are described some assumptions so as to define an unambiguous system. These assumptions will not be violated or changed during all the documentations.

1.5.1. General Assumptions

These assumptions will not be violated or changed during all the documentations. They concerning the global aspects of the system.

- Each user is not able to make nested rental of different cars during the same range of time.
- A single monthly invoice, which summarize all rentals and fees, is sent to the user in order to get the payment.
- Each car of PowerEnJoy has an interactive display inside the car.
- The employees of PowerEnJoy have the access to all the cars of the system.
- the employees are able to modify the status of each car if it is necessary.
- Plugging the car inside a special safe area for recharging the car is free of charge.
- The user, who terminated successfully the procedure of rental online, will not be able to cancel the rental.
- Safe Area are entire cities in order to facilitate access to parking, the system knows the set of Safe Areas.
- users can get of their car and decide not to end the ride in order to resume it later.
- Cars are automatically closed after the last passenger get off from it.
- The user who activates the “Money Saving Option”, before starting to drive, will get a discounts iff the car is plugged inside the special safe area suggested by the system.

1.5.2. Invoicing Assumptions

At end of each rental, the registered user will get a summary email containing all the information related the last rental.

The monthly invoice will show the debits during the month, specifying the use and any additional costs (for example, Penalties). In particular, after having carefully examined the case and ascertained the user's involvement, the system eventually will notify the user via email of the amount of penalty. The invoice will be issued in electronic format and can be downloaded from the user’s profile. Following this notification, the amount will be charged to the credit card or pre-paid credit card registered by the user.

These are example of penalties:

- administrative sanctions (e.g. due to traffic violations);
- service call-out (e.g. non-routine cleaning if you leave the vehicle too dirty; or if you leave it with the lights on);
- other (e.g. failure to report an accident, riding abroad);

If a user is not able to pay the monthly invoice will be banned from the system until he will be able to pay his debits back.

1.6. Constraints

Regulatory Policy:

-Privacy of all registered people must be granted. All the data will remain stored in the database of the system and will not be shared to third parts.

-Each registered user to the system has to provide a valid driving licence.

-Each user has to be associated with at least one valid credit card.

Hardware Limitation:

-Each user has to have a device able to know its position and send it to the system when unlocking the car.

-Each device has to be connected to the internet when interacting with the application.

-Each car has to maintain an internet connection in order to allow the system to lock and unlock it correctly.

-GPS presents on the devices used by user should be able to provide a decent position that enable the system to verify the distance between the user and the car-

-GPS presents on car should be able to place them in safe areas correctly.

-If the device of a user does not support the mobile application, the user can still use the application through the browser.

Interface with other application:

-The application take advantage of the services provided by Google Maps in order to calculate the route to arrive to the final destination and place correctly the safe areas on the map.

-The application will interface to a DBMS in order to store information about the users and the records of the rides

-The application will use services provided by credit cards companies to send the monthly invoice to users.

1.7. Reference documents

• Specification Document: PowerEnJoy Assignments AA 2016-2017.

• ISO/IEC/IEEE 29148 dated Dec 2011

2. Specific Requirements

2.1. Functional Requirements

Assuming that the assumptions stipulated in the paragraph [1.5] hold, and,

In order to fulfil the goals listed in paragraph [1.4], the following requirements

can be derived.

2.1.1. [G1] Users must be able to register to the system by providing their credentials and payment information

- [R1] Visitor must not be already registered to perform registration process.
- [R2] Username must be unique.
- [R3] An online form allows unregistered users to submit their credentials.
- [R4] The registered user can access to the system with a password.
- [R5] The system sends a password to the user after mail confirmation.
- [R6] The system sends a mail after the registration in order to verify the user's email address.
- [R7] The system before sending the confirmation mail must check the payment methods.
- [R8] The system provides a captcha inside the registration page in order to avoid bot attacks.
- [D1] User email address must be valid.
- [D2] Method of payment used for registration must be correct.

2.1.2. [G2] Registered users must be able to find the locations of available cars within a certain distance from their current location or from a specified address

- [R1] The system has the position of all cars.
- [R2] The system knows the level of charge of all cars.
- [R3] The system is able to acquire the user's position, if needed.
- [R4] The system provides a form where the user can insert a specified address and the radius of the search.
- [R5] The set of safe areas is associated with geographical coordinates.

- [D1] Each car has a GPS connected with the system.
- [D2] The system has a geographic map of the area.

2.1.3. [G3] Users must be able to reserve cars

- [R1] The system provides a form where the user can select one car for his rental.
- [R2] The system has a database in order to store all the rentals.
- [R3] The user is able to select only one car at once.
- [R4] Each car can be rented at once only by one person at maximum.
- [R5] The user’s last monthly invoice must be paid before the reservation.
- [D1] The user can rent only one car in the same range of time.
- [D2] All economic transactions end correctly
- [D3] It will be never possible that if a user takes more than one hour to pick-up his car, the system will not charge a fee of 1 EUR.
- [D4] The user can rent only one car for each rental.

2.1.4. [G4] A user must be able to enter in the car reserved by him when nearby

- [R1] The system can have the remote control of lock and unlock of all cars.
- [R2] The system must check the position of the user before unlocking the car.
- [R3] The application can send information from the user to the system.
- [R4] The user needs to provide his real and accurate position to the system.
- [D1] The user is connected to internet.
- [D2] The user has the access to the application.

2.1.5. [G5] The user is able to know the current charge of the ride

- [R1] Each car has an interactive display.
- [R2] The car is able to notify the system when the engine ignites.
- [R3] The system is able to calculate dynamically the amount of charge.
- [D1] Each car provides a start/stop button.

2.1.6. [G6] The system is able to calculate the final cost of the rental

- [R1] The system is able to calculate discounts
- [R2] The system is able to know the number of passengers inside the car.
- [R3] The system is able to know the number of cars inside each special safe area.

2.1.7. [G7] The system is able to know when a ride ends

- [R1] The car is able to understand if a user leaves the car.
- [R2] The system is able to understand if the car is parked in a safe area.
- [R3] The car is able to communicate with the system.
- [R4] The system is able to lock the car automatically after it stops charging the user.
- [D1] There is always a connection between the car and the system.

2.1.8. [G8] The system is able to manage the cars

- [R1] The system can set a car unavailable.
- [R2] User can notify the system in case of malfunctioning.
- [R3] The system knows the position of each car and the level of their batteries.
- [R4] The system, through the employees, can move the cars in case of necessity and recharge it on site.

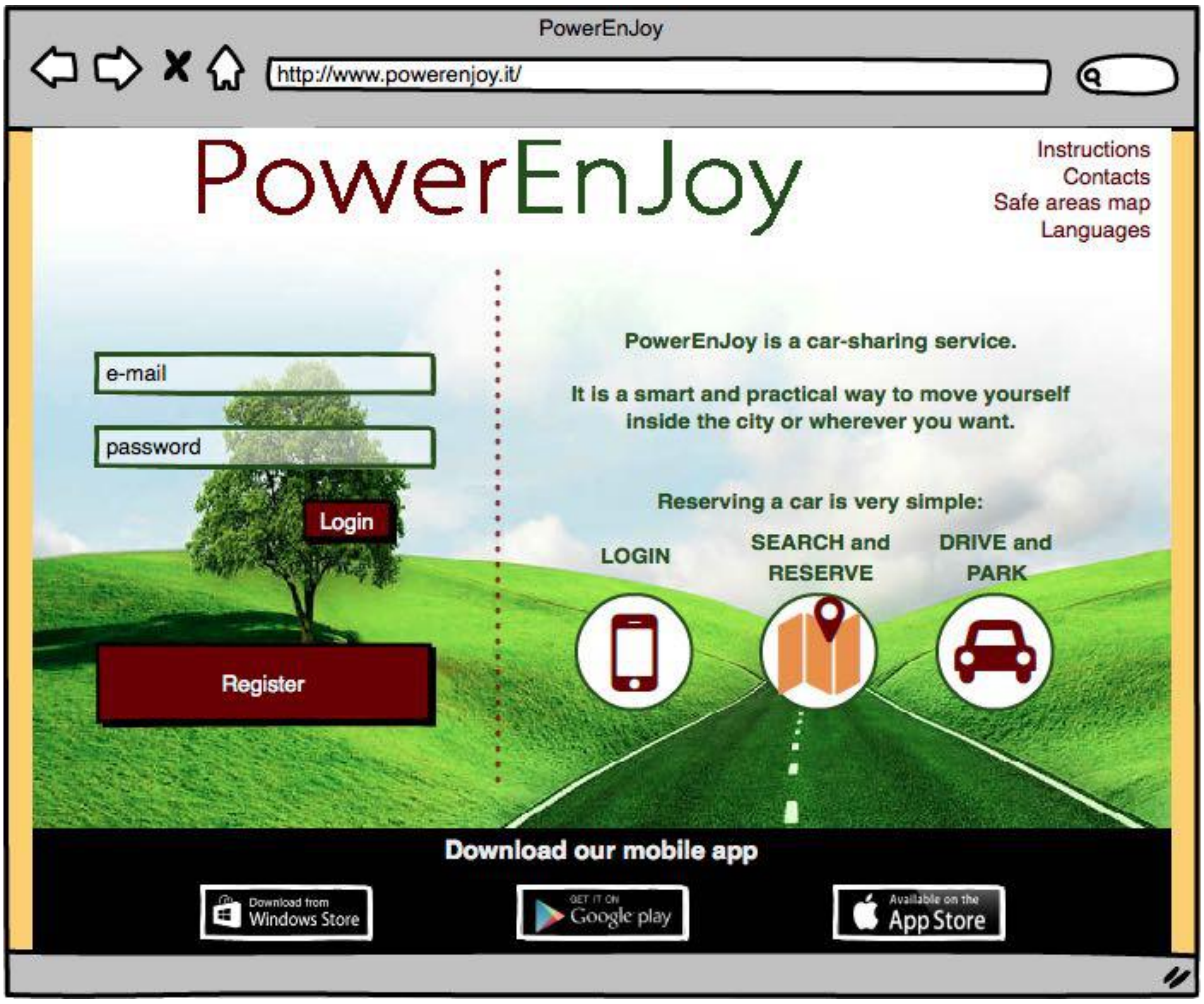
2.1.9. [G9] The user is able to update his profile and payment methods

- [R1] The system provides forms in order to update user’s information.
- [R2] The system checks the new payment methods inserted.

2.2. Non-Functional Requirements

2.2.1 User Interfaces: Login

The mock-up above shows the home page; it appears as soon as the user reach the website. The home page allows the users not only to login inside the website but also to register a new profile. A small description of the features of this car-sharing service is located on the right of the page.



PowerEnJoy is a car-sharing service.

It is a smart and practical way to move yourself inside the city or wherever you want.

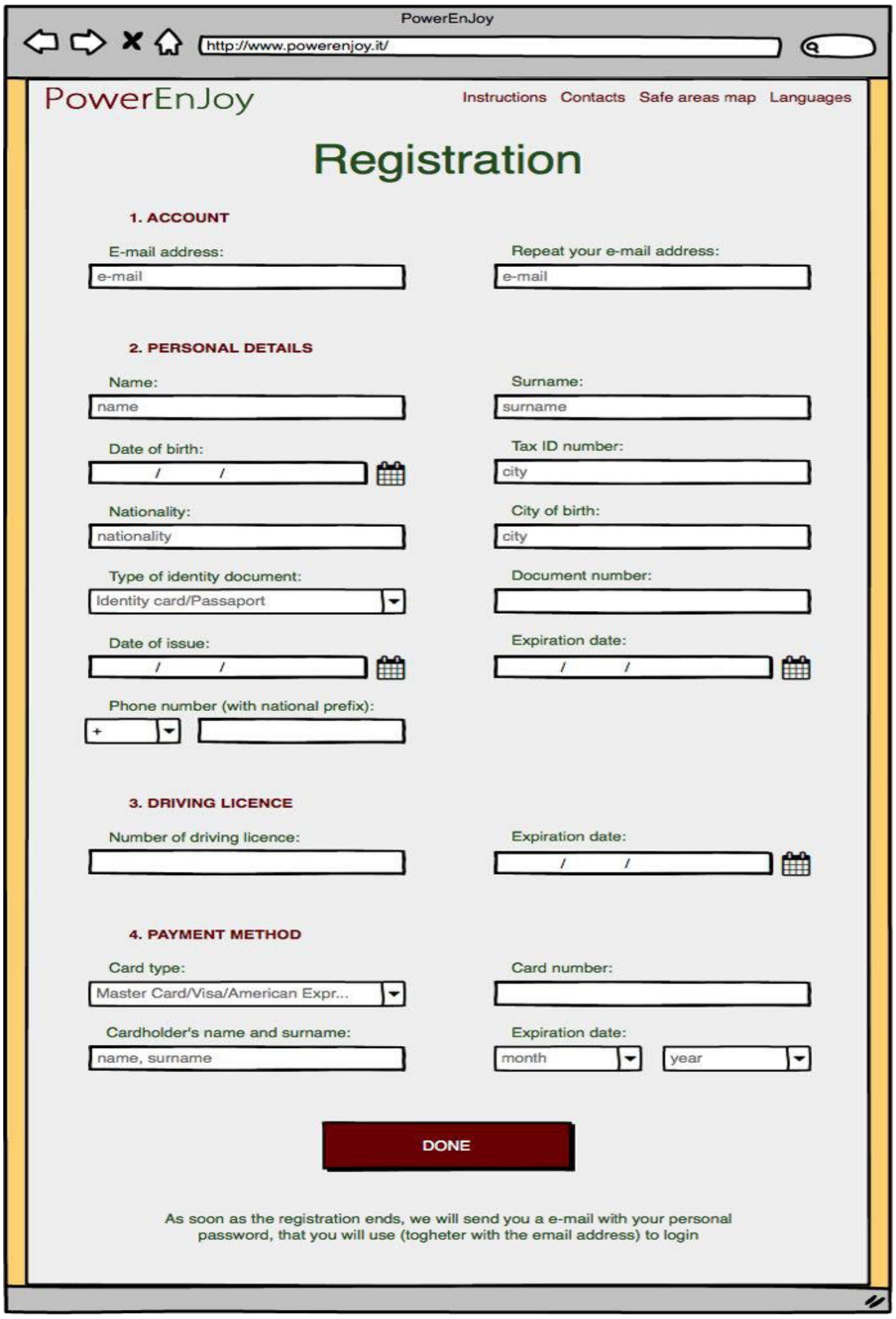
Reserving a car is very simple:

Download our mobile app

Figure 1

2.2.2 User Interfaces: Registration

As mentioned previously the unregistered user can access to the registration page from the home page. This page is composed of a set of forms, the unregistered user needs to fulfil all the forms in the page in order to register his profile.



PowerEnJoy

http://www.powerenjoy.it/

PowerEnJoy

Instructions Contacts Safe areas map Languages

Registration

1. ACCOUNT

E-mail address: e-mail

Repeat your e-mail address: e-mail

2. PERSONAL DETAILS

Name: name

Surname: surname

Date of birth: / /

Tax ID number: city

Nationality: nationality

City of birth: city

Type of identity document: Identity card/Passaport

Document number:

Date of issue: / /

Expiration date: / /

Phone number (with national prefix): +

3. DRIVING LICENCE

Number of driving licence:

Expiration date: / /

4. PAYMENT METHOD

Card type: Master Card/Visa/American Expr...

Card number:

Cardholder's name and surname: name, surname

Expiration date: month year

DONE

As soon as the registration ends, we will send you a e-mail with your personal password, that you will use (togheter with the email address) to login

Figure 2

2.2.3. User Interfaces: Search & Reserve

After the registration (or login) the user can search and reserve a car in this web-page. It is possible to find a car in two different ways, from their current position (with the user's GPS) or from a specified address. The user can also define the radius in which he is able to find cars.

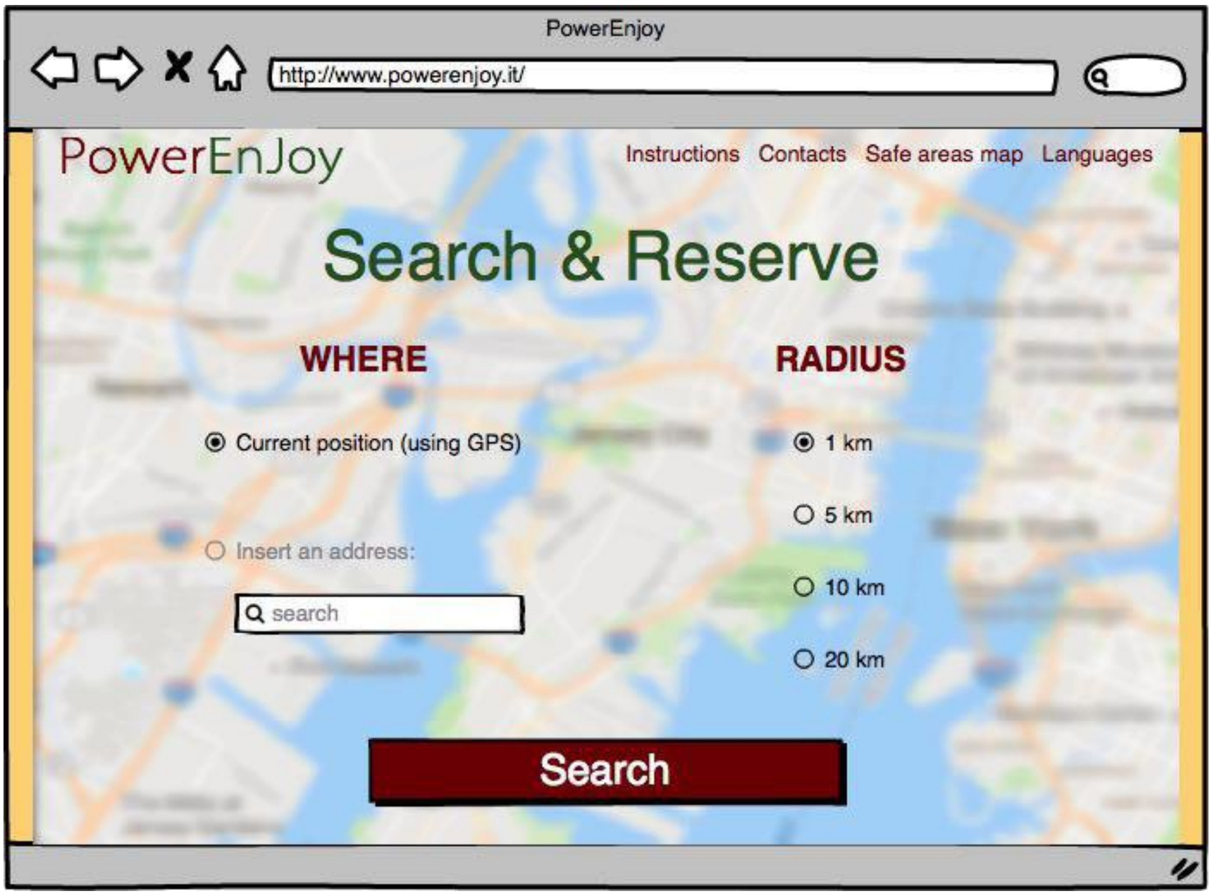


Figure 3

2.2.4 User Interfaces: map and cars

The system sends to the user's client the information which he was looking for. The map contains all the information that the user needs, inside the user can see all the cars inside the radius of the point he has defined but also his position. The labels inside the map spot the position of cars. On the right of the page, you can see a small legend in which the user can see the meaning of colours of labels.

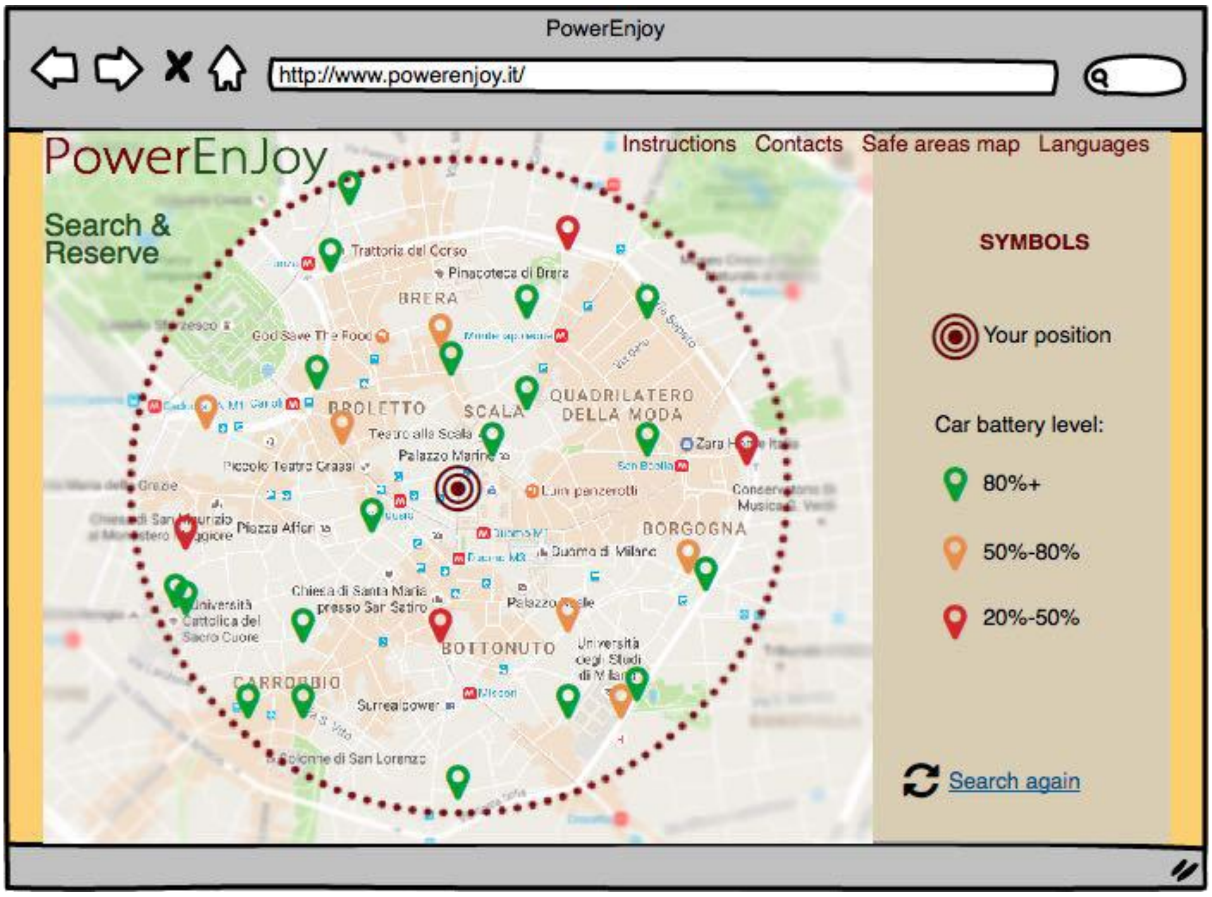
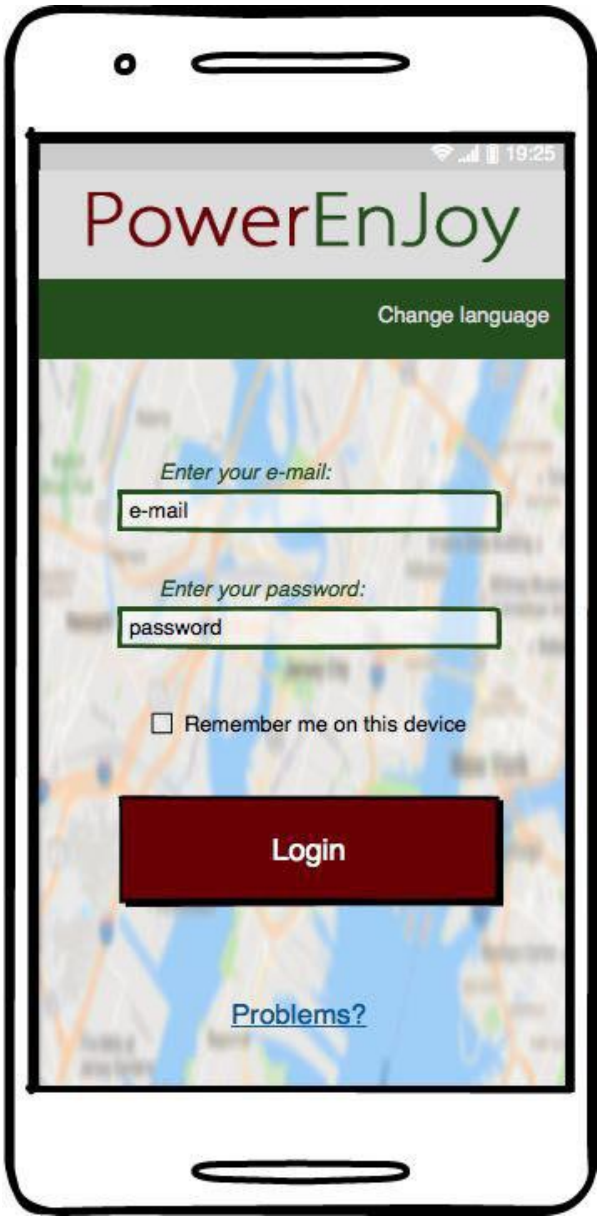


Figure 3

2.2.5. User Interfaces: mobile application login and settings

These two mock-ups represent the login page and the settings panel for mobile application. User can access to the features of the application only after the login.



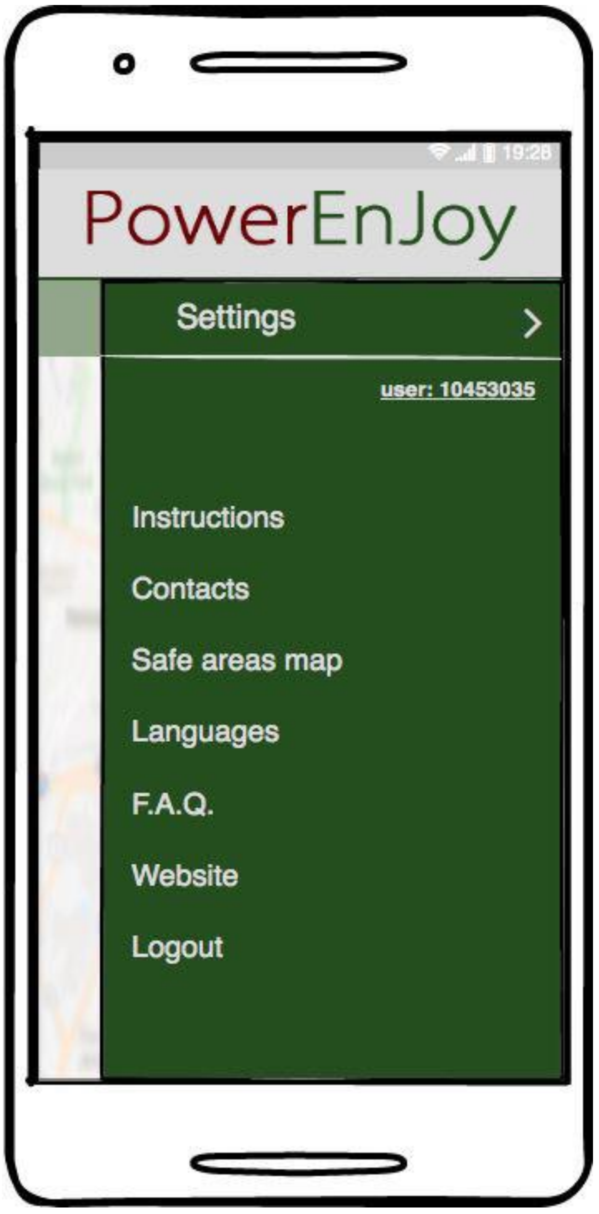


Figure 4

Figure 5

2.2.6. User Interfaces: mobile application, rental stage

After login the user can rent a car. In the picture on the right the user is able only to rent a car, but after the application unlock the other functionalities (picture on the left). The functionalities of application are:

- Reserve a car: the user can rent a car
- Car's position: find the position rented car on the map
- Unlock the car: iff the user is nearby a car he is able to ask for unlocking the car.
- End the rental: iff the user is out of the car and the car is parked in a safe area he is able to end the rental of a car.

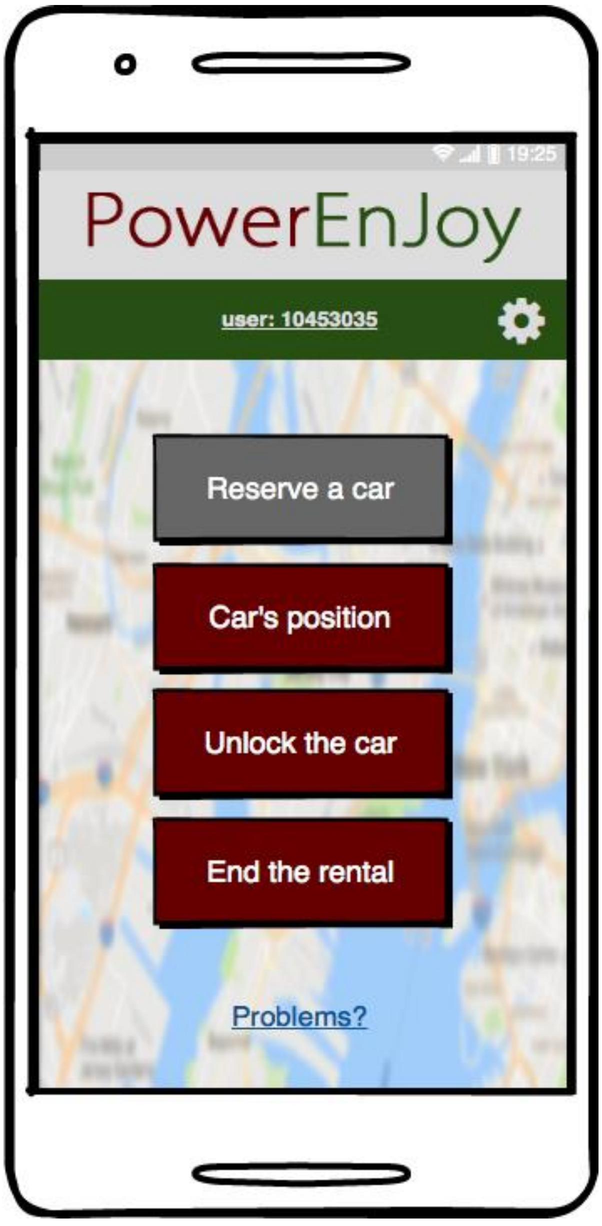


Figure 6

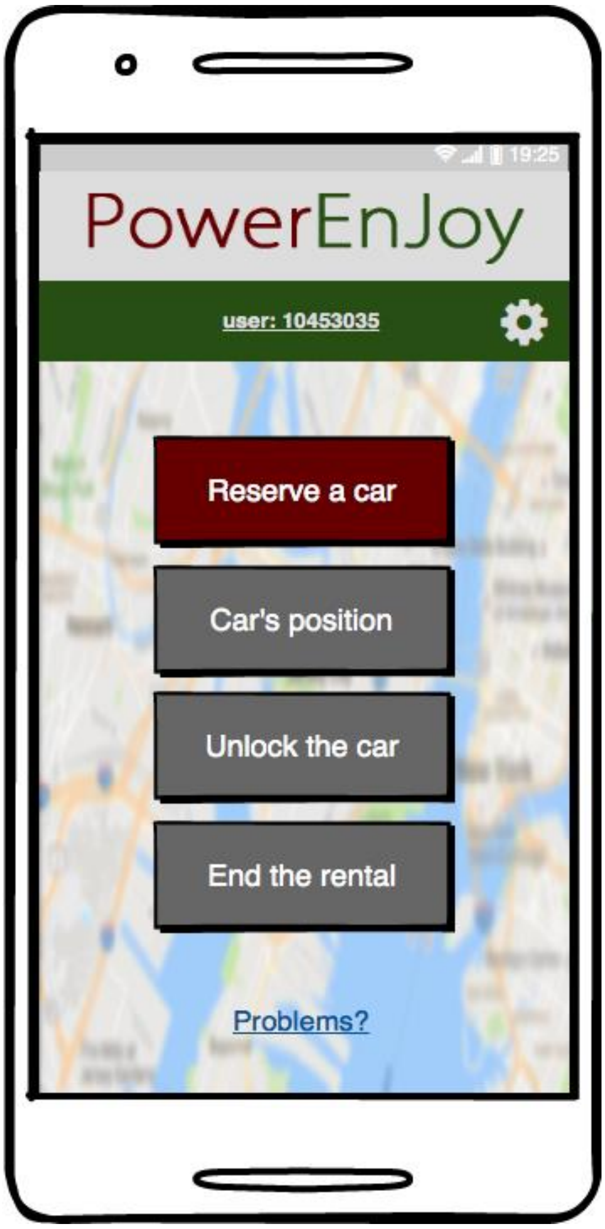
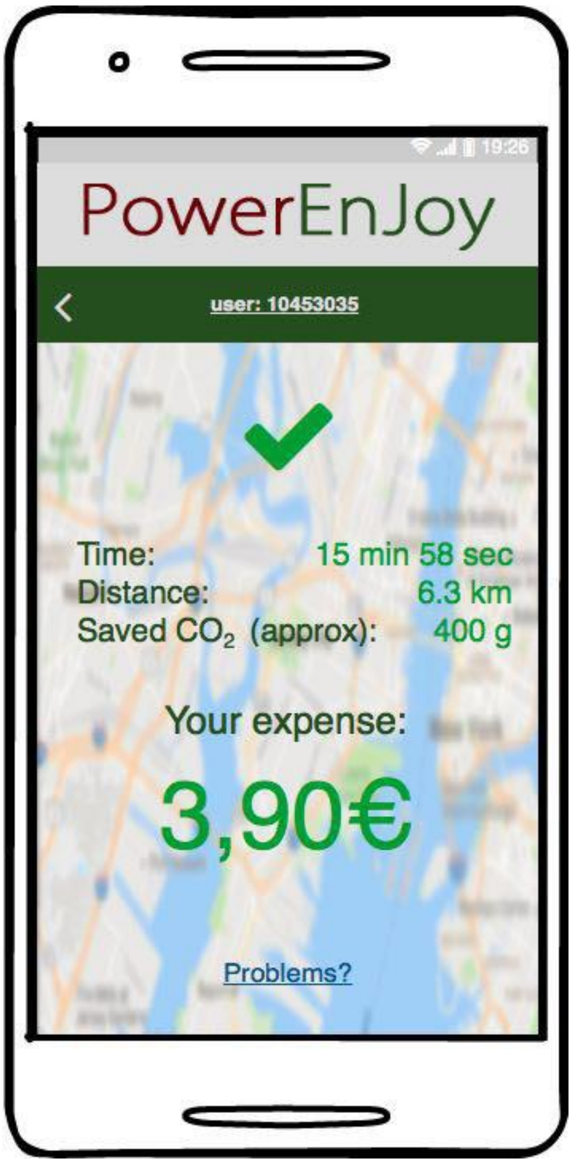


Figure 7

2.2.7. User Interfaces: mobile application, end of rental

At the end of the rental, the user receives a notification on his application. The notification contains all the information related to the last ride. The user can know each detail of his rental like an invoice (time, distance and expense).



Time: 15 min 58 sec
Distance: 6.3 km
Saved CO_2 (approx): 400 g

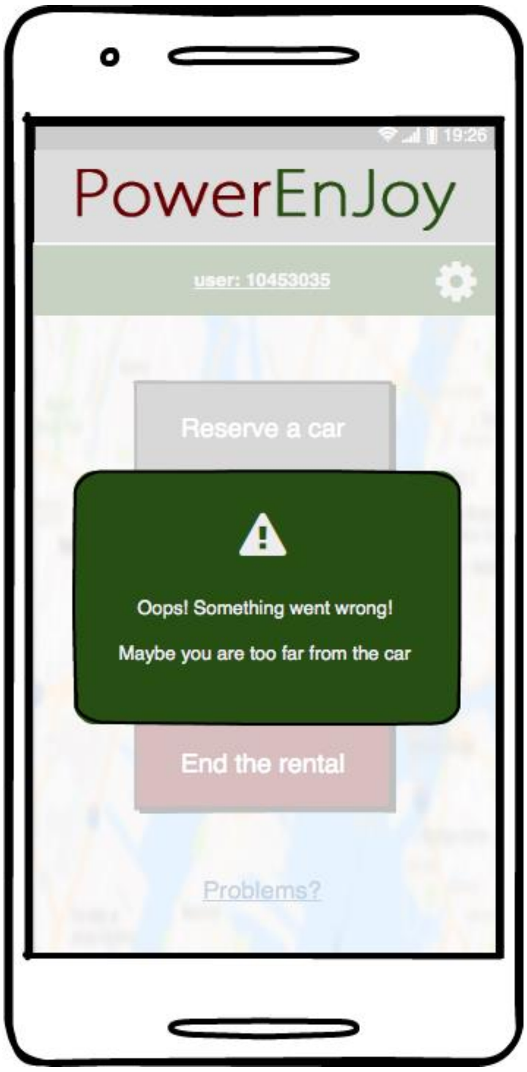
Your expense:

3,90€

Figure 8

2.2.8. User Interfaces: mobile application, unlock the car

This error will appear iff the user tries to unlock the car but he is not nearby the car.



Oops! Something went wrong!

Maybe you are too far from the car

Figure 9

2.2.9. Usability

The choice of extending safe areas to entire cities facilitates users while looking for a park, in fact, in this way they can use all the parking opportunities offered by the city. Moreover, the minimalism of the interface of the application allows user to speed up the processes of renting a car and lock/unlock it. The application will be released in the three main app market (Windows Store, App store, Google Play) and allows users to manage multiple accounts on the same phone. In order to avoid people stuck in the cars with the battery of their phone at 0% we provide USB power plugs within the cars so that users can recharge their phone and end the ride through the application.

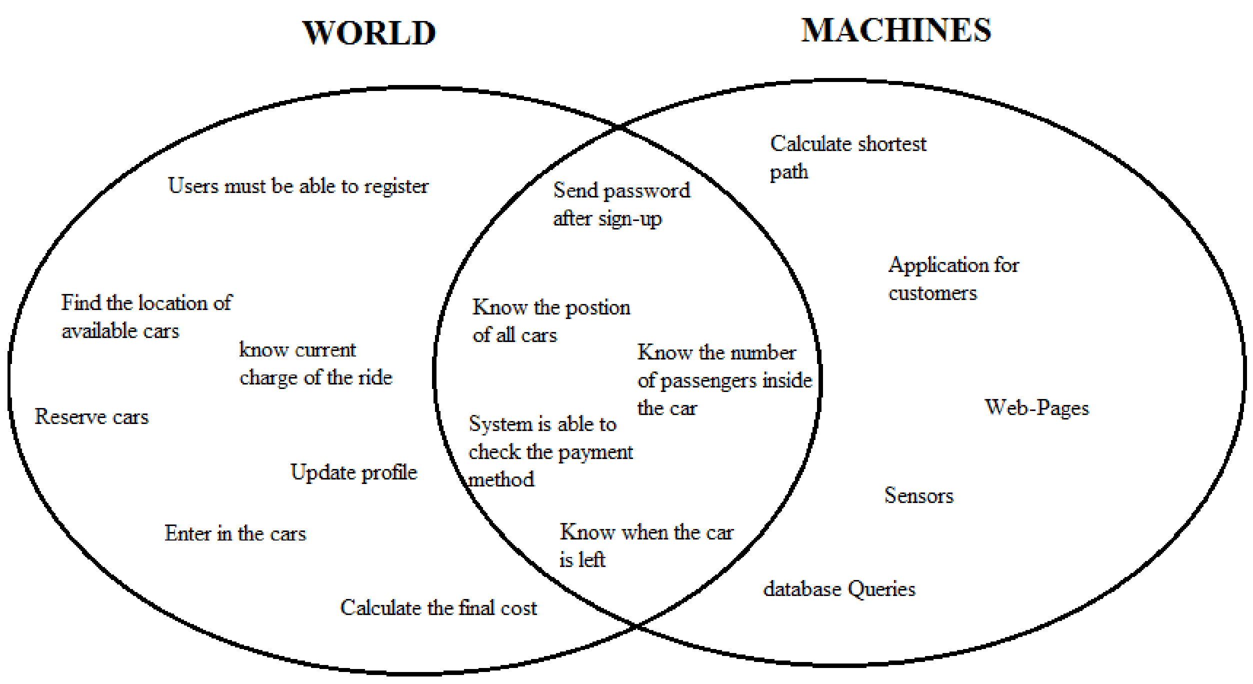
2.2.10. Privacy requirements

The system will protect users' personal data by storing them in safe serves. In particular, Login credentials are encrypted by using hash functions.

All the information related to rides will be private and won't be shared.

2.3. The world and the Machine

For an overall analysis of the PowerEnJoy environment we use "The world and the machine" modelled by M.Jackson & P.Zave. This approach allows us to identify the elements inside the domain that interact with the application ("World"), elements which need to be developed and create ("Machine") and the intersections that are all the variables know by the machine in order to perform its functionalities.


WORLD
MACHINES
Users must be able to register
Find the location of available cars
know current charge of the ride
Reserve cars
Update profile
Enter in the cars
Calculate the final cost
Send password after sign-up
Know the postion of all cars
Know the number of passengers inside the car
System is able to check the payment method
Know when the car is left
Calculate shortest path
Application for customers
Web-Pages
Sensors
database Queries

3. Software Design – UML

3.1. Actors definition

In the system there are four main actors that can be identified:

-Unregistered user: is the one that would like to use the system but is not registered yet.

-User: Is the one that take advantage of the services offered by the system.

-Car: is the object composed by a set of sensors and antennas that allows user to move around the city and manage reservations and rides.

-Employee/Operator: is the person that support users during their rides; He provide guidelines in case of incidents or problem and is able to end rides and set car unavailable when under maintenance.

3.2. Possible scenarios

Scenario 1

Miss B is an actress and she knows that in few days she will probably have an audition for the next film of Sorrentino set in Rome, where she lives. So she wants to register herself on the PowerEnJoy system. She opens the website and goes in the 'registration' area: she inserts name, surname, date of birth, email, ID code, driving license and references to a payment method, after that she receive back her new personal password. Now she can log in and reserve a car whenever she wants.

Scenario 2

Mr. and Mrs. C are just married and they have just bought a new home. Today they want to buy new furniture at the near Ikea store. Unfortunately, this store is 3km far from the safe area of Milan, so they are now reserving a car (which is 5 minutes far from their home) and later they will pick the selected car and then they will drive until the store, where, they will stop the car. However, since the store is outside from the safe areas they cannot end the ride and therefore when they will go shopping, they will be paying, too. When they will finish, they will come back and park inside the safe area.

Scenario 3

Mark is an Australian tourist that is visiting Milan, with his family, for the well-known fashion week. Unfortunately, the train from the airport arrives in the Milan central station at 1 a.m. and therefore he has no idea about how reach the city centre. He would like to avoid the high fees of a taxi and at the same time respect the environment; that is why he chose PowerEnJoy. After logging in on the web site he looks for the nearest car and gets the one parked outside from the station and discovers the GPS navigator service present on the car. This engaging feature allows Mark to find the closest path to the centre and at the same time the closest power grid station where parking. Mark parks the car at the special safe area and connect the car to the power plug. The system notice that and consequently apply the 30% discount on the ride and moreover an additional 10% discount is added due to the presence of more than two passengers detected by the sensors.

Scenario 4

Massimiliano is arriving by train to Rome and decides to plan his visit by booking a car on the PowerEnJoy system. However, due to a problem of an engine, the train arrives at the station with a delay of one hour. Therefore, Massimiliano is not able to get the car within one hour from the reservation that expires. The user is charged of a fee of 1 euro and the car becomes available again.

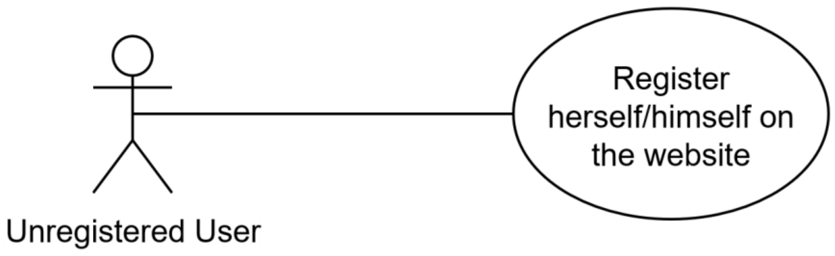
Scenario 5

Mr. Jones is a rich tourist, who loves cars well designed. During the month of October, he needed to reach several times the Milan Cathedral situated in the centre of Milan. The access to the historical centre of Milan is limited by the Congestion Charge area (Area C) from Monday to Friday. Motorcycles and scooters, electric vehicles are exempted from payments. So Mr. Jones used PowerEnJoy. Mr. Jones is also a cheapskate person, in order to save money, he left the car with more than 50% of battery because the system applies a discount of 20% on the last ride.

Scenario 6

Mrs. Robinson’s grandparents lives in Italy. Twice a year she wants to visit his grandparents, but they live far from city centre, so she can’t use subway or buses to reach his grandparents. For this reason, she uses PowerEnJoy. They live 3 km from the nearest power grid station and far from the railway station. So the system charges 30% on her the last ride more to compensate for the cost required to recharge the car on site.

3.3. Actor: Unregistered user



Use case: Register herself/himself on the website

Description:

An unregistered user, that is someone who doesn't have an account yet, can submit his data, in the website, in the registration page.

After the system has verified and saved all the information, the guest becomes a Registered User, and after he will receive his personal password

Actors:

Unregistered User

Input Conditions:

Null

Output Conditions:

Unregistered user successfully ends the registration process, and becomes a Registered User. From now on he/she can log in to the application or the website using his/her email address and password, then can access to peculiar functionalities.

Events flow:

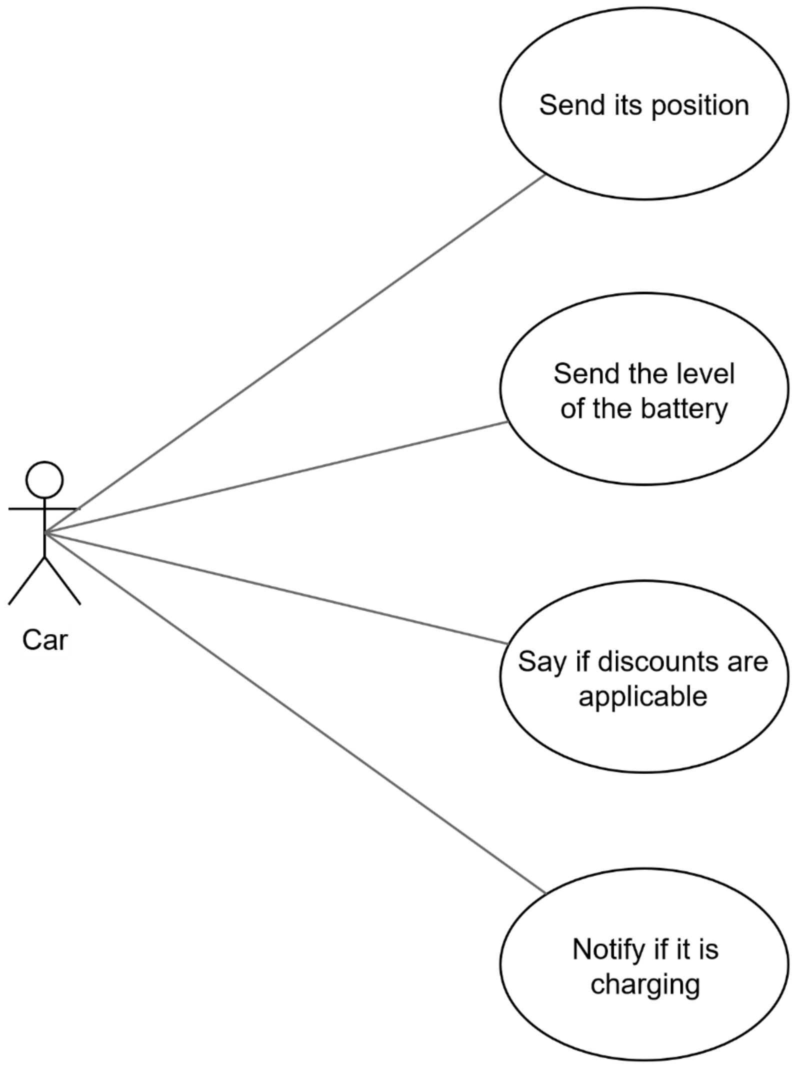
- Unregistered User on the home page clicks on the “register” button to start the registration process;
- Unregistered User fills all the fields;
- Unregistered User clicks on “DONE” button;
- The system will verify all the inserted data and then will save it in the database;
- The system will generate a password for that new user, which will be sent to him/her.

Exceptions:

- The guest has already registered;
- The email inserted is already associated to another user;
- One or more fields are empty;
- At least one of the inserted data are not valid;

All the exceptions cause the application to notify the error to the user with an alert window. The application then come back to the registration form, and the Event Flow will restart from the filling step.

3.4. Actor: Car


Send its position
Send the level of the battery
Car
Say if discounts are applicable
Notify if it is charging

Use case: Send its position

Description:

A Car, can notify the system with its position thanks to his integrated GPS service.

Actors:

Car

Input Conditions:

Request of its position (system)

Output Conditions:

The system now has the selected car position (couple of coordinates).

Events flow:

- The System connects to a specific car;
- The System asks for its position;
- Car takes the couple of coordinates with the GPS;
- Car sends this information to the system.

Exceptions:

- Car sends at least one unreal coordinate;
- GPS is not working

This exception causes the system to notify the car with an error message, and after that the system keeps on trying with the position request, and it will restart from the asking step. If the process fails again an employee

Use case: Send the level of the battery

Description:

The car notifies the system with the level of its battery.

Actors:

Car

Input Conditions:

Request of battery level (system)

Output Conditions:

The system has percentage of the level of the car’s battery.

Events flow:

- The System connects to a specific car;
- The System asks for its battery level;
- Car send this information to the system.

Exceptions:

- Car send an unreal value;

This exception causes the system to notify the car with an error message, and after that the system keeps on trying with the battery level request, and every time it will restart from the asking step. When the level is under 5% the car automatically sends this message to the system in order to notify it that the car will stop soon.

Use case: Possible discount for more than one passenger

Description:

The car can notify the system with the availability of the discount for more than one passenger for at least the half of the rental.

Actors:

Car

Input Conditions:

Request by of the possibility of applying a discount (system)

Output Conditions:

The system now has the average number of people, who have done the travel and if it is at least two a discount will be applied

Events flow:

- Car begins to calculate the value at the beginning of each ride
- The car updates the current number of passengers every time the sensor detects a new person
- The car calculates the final value as the weighted average of passengers during the ride over the time.
- The System connects to a specific car;
- The System asks for the availability of the discount;
- Car sends this information to the system.

Exceptions:

Car is not able to provide a valid number

This exception causes the system to notify the car with an error message, and after that the system keeps on trying with the position request, and every time it will restart from the asking step. If the process fail again the system will not apply the discount for the number of passengers

Use case: Notify if it is charging

Description:

A Car notifies the system with a charging message.

Actors:

Car

Input Conditions:

The plug of the power station has been inserted in the car, and the charging phase started.

Output Conditions:

The system receives the message, that contains the power station's code, too. Therefore, now the system knows if the car is charging, in which station and also in which SpecialSafeArea.

Events flow:

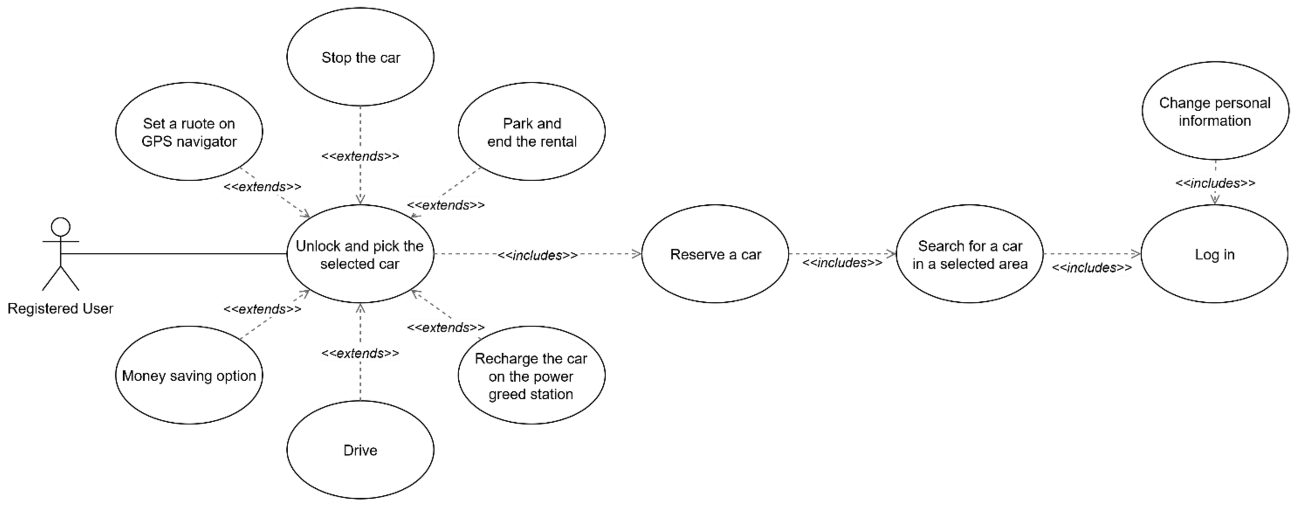
- The car notice that is in charge;
- It takes the plug information as soon as it starts charging;
- The car connects notify the system with a charging message.

Exceptions:

- Car send an unreal value for the plug's code;

This exception causes the system to notify the car with an error message, and then the car starts this process again (starting from the second phase)

3.5. Actor: registered user



Use case: Log in

Description:

A user already registered can provide his credential (email address and password) to log in to the application, becoming a Registered User and thus gaining some privileges.

Actors:

Registered User

Input Conditions:

The person that wants to log in must be already registered to the system.

Output Conditions:

The user is now logged into the webpage or the app as a Registered User and may do lots of actions.

Events flow:

- The user fills in the log in the form present in the home page (on the web page or on the mobile application).
- The system verifies the inserted credentials, and if they are correct, let the Registered User to access.

Exceptions:

- Credentials are not valid;
- Email address and password do not match.

In case of these exceptions, an alert window will be shown and the access denied, remaining on the home page, and giving the guest the possibility to try again.

Use case: Search for a car

Description:

An user, already logged in, can search for a car writing an address or using his GPS position and selecting a radius for the research.

Actors:

Registered User

Input Conditions:

The user must be already logged in. For the GPS position the user must make the research with a device that is able to send the position.

Output Conditions:

The user can select the desired car form the ones proposed

Events flow:

- The user fills in the log in form already present in the home page (on the web page or on the mobile application);
- The system search for all the car in that selected area and sends all of the resulting available cars to the user;

- The user now sees all the car and the battery level, in the area he/she selected.

Exceptions:

- User wants to provide the position via GPS, with a device that cannot provide a GPS position;
- User provides an invalid address.

In these cases, an alert window will be shown and the search starts again.

Use case: Reserve a car

Description:

A user, after a valid research, can reserve one of the available cars.

Actors:

Registered User

Input Conditions:

The user must have already done a valid research, which have returned at least one car.

Output Conditions:

The user has an active reservation, that expires in one hour.

Events flow:

- User selects one of the available cars proposed by the system and reserves it;
- The system creates a reservation with the user and the selected car;
- User has a reservation, that expires in one hour.

Exceptions:

- User wants to reserve a car while he/she has already one (underway).

In these case, an alert window will be shown, and returns on the home page.

Use case: Money Saving Option

Description:

In order to get discounts, registered users can activate “Money Saving Option” before defining the destination on the interactive display.

Actors:

Registered user

Input Conditions:

The registered user wants to get a discounts of 40%.

The registered user has just picked up the car of his rental.

Output Conditions:

The car is inside the special safe area suggested by system and plugged into a power grid, so as to have a better distribution of all the cars.

Events flow:

- The registered user picks up a car and activate “Money Saving Option”;
- The user sets the destination;
- The system analyzes the uniform distribution of cars in the city, the user’s destination and the availability of power plugs inside special safe area;
- The system defines the optimal route for the user;
- The system checks if at the end of rental, the car respects all the output conditions;

Exceptions:

The user leaves the cars in the wrong position;

After this exception, the system will not provide the discount to the final invoice of this rental. The system will notify that inside the final monthly invoice which summarize all costs of rentals.

Use case: Unlock and pick the selected car

Description:

A user, while his reservation is valid, is able to unlock the car and pick up it.

Actors:

Registered user

Input Conditions:

A reservation must be valid and the user must be nearby the rented car (inside a radius of 20 meters from the car).

Output Conditions:

The car is unlocked and the user is able to enter inside the car.

Events flow:

- The user presses the button on his mobile application when he is less 20 meters far from the rented car;
- The system receives the request;
- The system checks the distance between user and car;
- The system unlocks the car and the ride starts;
- The car is now unlocked and the user can enter inside.

Exceptions:

- User tries to unlock the car when is more than 20 meters far from the car;

In this case, a negative message will be show on the mobile application, telling the user that he must be nearby the car.

Use case: Set a route on a GPS navigator

Description:

A user, whenever he wants, after the ride is began, can set a route on the car GPS, even though there was a previous planned route (which will be automatically deleted).

Actors:

Registered user

Input Condition:

Setting a GPS route can be done extending the unlock action.

Output Condition:

The user has a route on GPS navigator.

Events flow:

- The user writes an address on the car’s navigator;
- The car computes a route, with the provided data, and shows it on the interactive display.

Exceptions:

User provides an invalid address.

In these cases, an alert window will be shown and the search starts again.

Use case: Stop the car

Description:

The user is able to exit from the car, whenever he needs. The system locks the car when no one is inside the car and engine is turned off. In the meantime, the ride still goes on, so the user will pay as a normal use.

Actors:

Registered User

Input Conditions:

The action of stopping the car can be done extending the unlock action.

Output Conditions:

The user’s rented car is locked by system.

Events flow:

- The user exits from the car;
- The car sends that no more passengers are present on the car;
- The system locks the car;

Use case: Park and end the rental

Description:

A user, whenever he wants during his rental, can park and end the rental.

Actors:

Registered user

Input Conditions:

Parking and finishing the rental can be done extending the unlock action.

The user must have previously stopped the car in one of the Safe Areas.

Output Conditions:

The system locks the car and set it available again. It also finishes the user’s ride.

The user is again able to make a new reservation.

Events flow:

- The user stops the car in a Safe Area;
- The user presses the button to end the rental;
- The system verify that the car is stopped inside one of the Safe Areas.
- The system stops charging the user;
- At the end, the system analyzes the data from car in order to know if there are some applicable discounts.

Exceptions:

- User tries to end the rental inside the car;
- User tries to unlock the car, even is already parked;
- User tries to park the car, even though he/she has not stopped the car in a Safe Area;

In these exceptions, an alert window will be shown on the app and after the home page appears.

Use case: Drive

Description:

A user, whenever he wants, after he has picked up the car he is able to drive and move the car.

Actors:

Registered user

Input Conditions:

Driving the car can be done extending the unlock action.

Output Conditions:

The user can move the car wherever he wants.

Events flow:

- The user turns on the engine of the car;
- The user is able to drive;

Use case: Charge a car in a power grid station

Description:

Users are able to plug his rented car inside any power grid station. Power grid station inside the special safe area are free of charge, but when an external power grid station is used in order to recharge the car the costs are all on user’s expense. The ride cannot be terminated when the user start charging the car in an external power grid station.

Actors:

Registered user

Input Conditions:

Recharging the car on a power grid station can be done extending the unlock action.

Output Conditions:

The car is in charge.

Events flow:

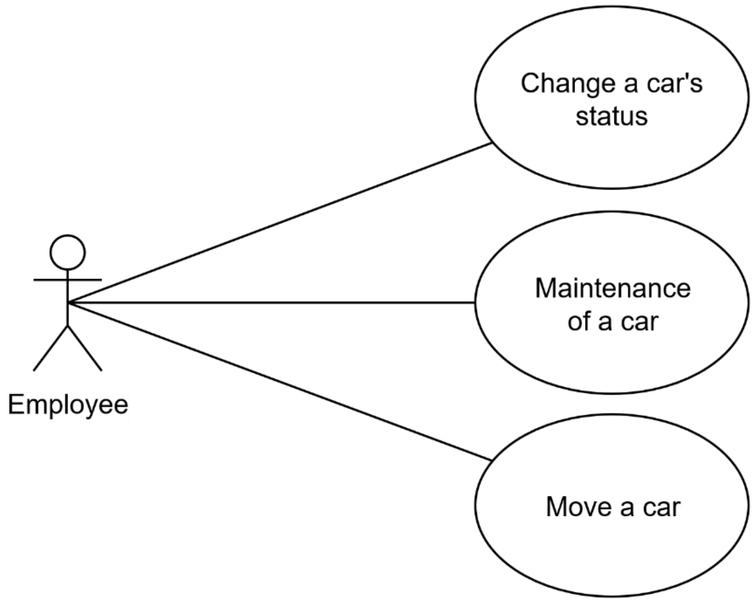
- The User stops the car;
- The User connect the car with the wire of the power grid station;
- The car starts charging;
- The user has to pay if he used an external plug.
- User can disconnect the car and continue his rental.

Exceptions:

- User tries to charge the car after he/she had pushed the button to end the rental

In this case, the car does not allow to the user to charge, because the system has locked the opening for the plug.

3.6. Actor: Employee



Use case: change a car’s status

Description:

An employee can change manually the status of a car.

Actors:

Employee

Input Condition:

Some problems appear.

Output Condition:

The car has now a different status.

Events flow:

- One problem appears;
- Employee changes the car status manually;
- The system checks the old and the new statuses;
- The car has a new status.

Exceptions:

- Employee set a new status that is the same as the last one.

With this exception the system undoes all the operation that the employee has done, letting him start from the begin.

Use case: maintenance

Description:

An employee can manage the maintenance of a car.

Actors:

Employee

Input Conditions:

A car is damaged or a problem occurred

Output Conditions:

The car is now available.

Events flow:

- Employee receives a call from a user or notify some problem with a car
- Employee set the car not available from his reserved area of the application.
- Send guidelines to follow to the user.
- If necessary, he recharges the car on the spot

Use case: change a car's position

Description:

An employee moves a car in order to improve their distribution on the map.

Actors:

Employee

Input Condition:

The cars are not well distributed on the map.

Output Condition:

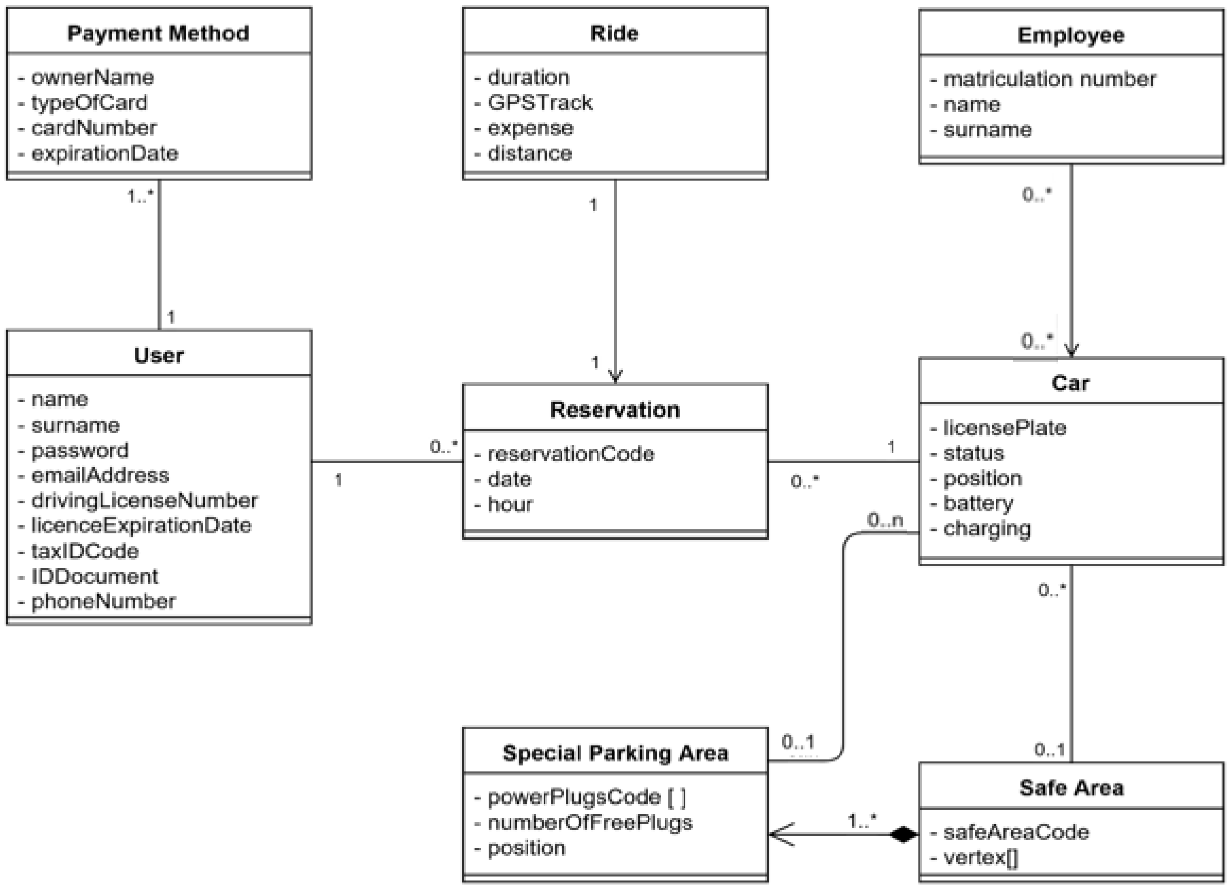
Cars are now usable again and placed in a best distributed position.

Events flow:

- The system asks to an employee to move some cars;
- All these cars are set not available
- The employee discovers that a car need to be moved to another position.
- The employee goes to take the car and move it
- The employee sets the car available again

3.7. Domain Model

3.7.1. Domain Class Diagram



This Class Diagram shows the interaction between the classes of the system. First of all, there is the User that can have more than one Payment Method, that are different cards, among which at least one must be valid.

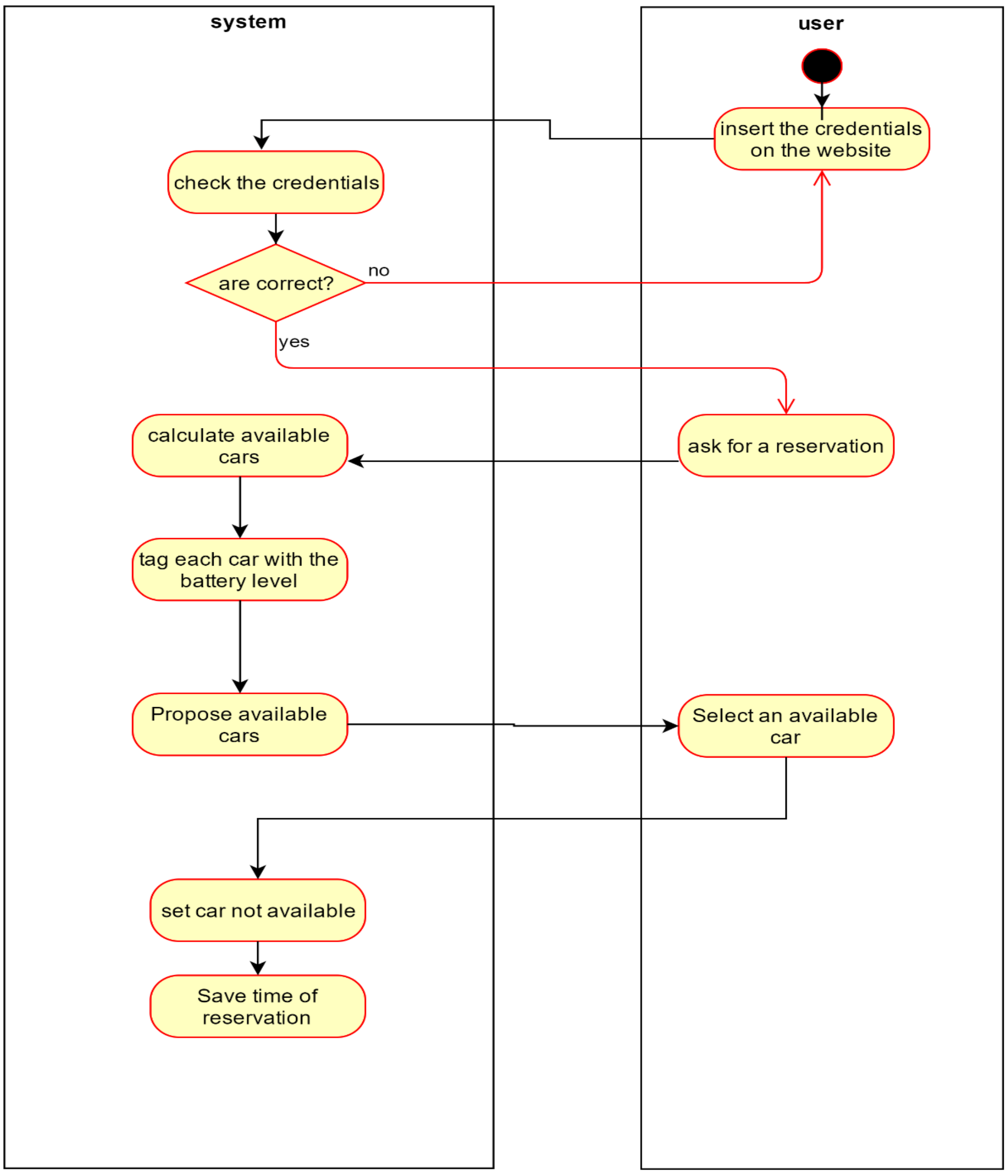
One user, if his driving licence is not expired, can make a Reservation, selecting one Car among all the available cars. User can make more than one reservation, but each reservation must be done after that his last ride is finished. A car can be in more than one reservation, but not at the same time. Each reservation has exactly one correlated Ride, that shows even the case that the user does not reach the car in less than one hour (the duration will be 0 minutes and the expense will be 1 EUR). At the end of the rental, the system computes the final amount to pay, considering all the discount or penalties.

Car may be parked, but only in a Safe Area, which can contain one or more Special Parking Area (Special Safe Area). All the Special Parking Areas have one power grid station with a certain number of plugs and each plug has a univocal code. If the user parks the car in a Special Parking Area, inside a Safe Area, he can charge the car in the power grid station, if there is at least one free

3.7.2. Main dynamics of the system

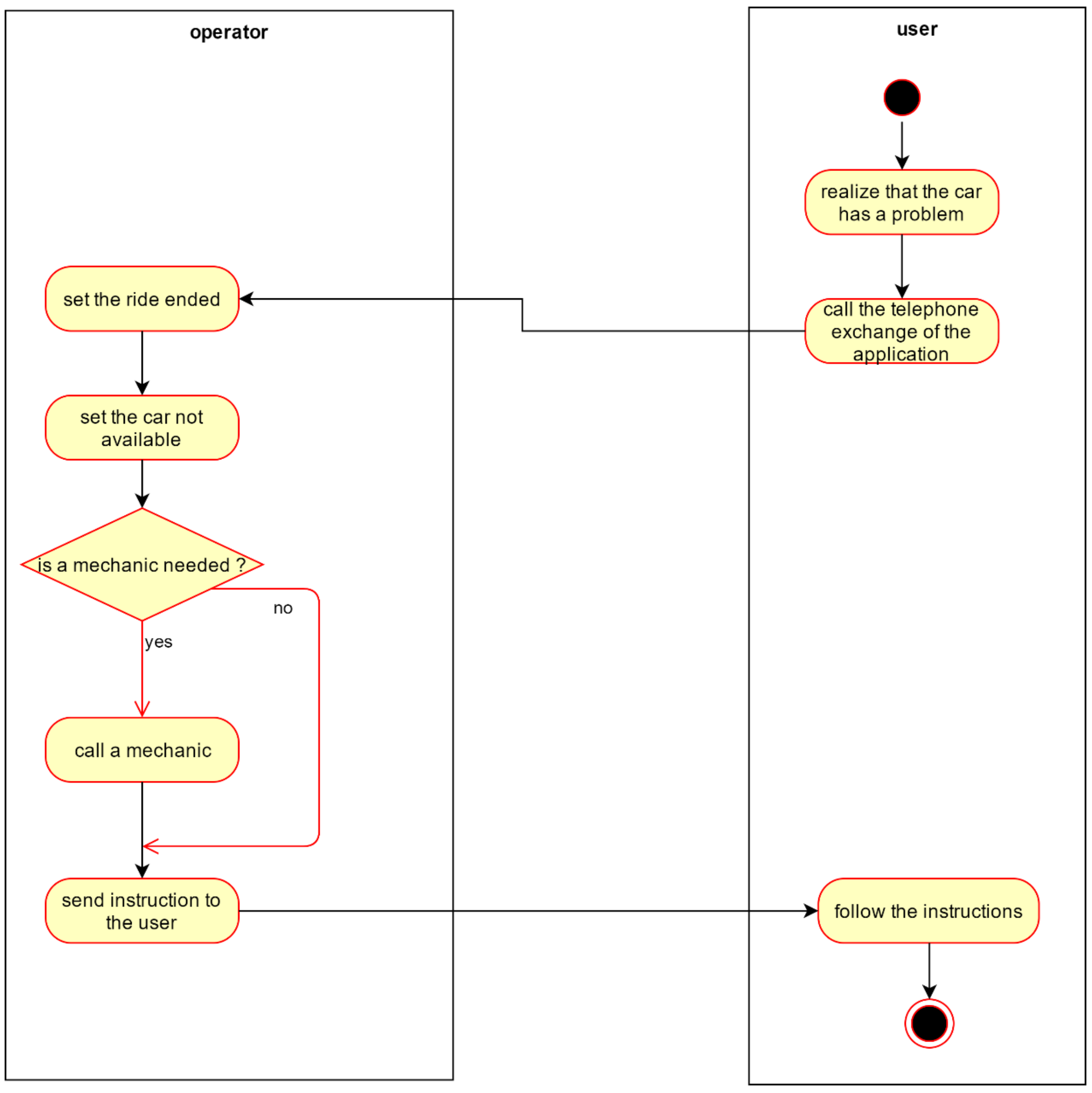
Reservation Activity Diagram

The diagram illustrates the process of car reservation performed by a user. Firstly, the user logs in the website by providing his credentials and then asks to the system the available cars (using GPS position); If the credentials are correct the system proposes a map containing all the available cars and the user has the possibility to choose one of them. Each car is labelled with the level of the charge and all the car that are rented or with a level of the batter below 20% are not shown. The user select the most suitable car based on the position and at this point, the reservation is completed, the car is set to unavailable and the time of the reservation is stored in order to check that the user collects the car within one hour.



Operator tasks Activity Diagram

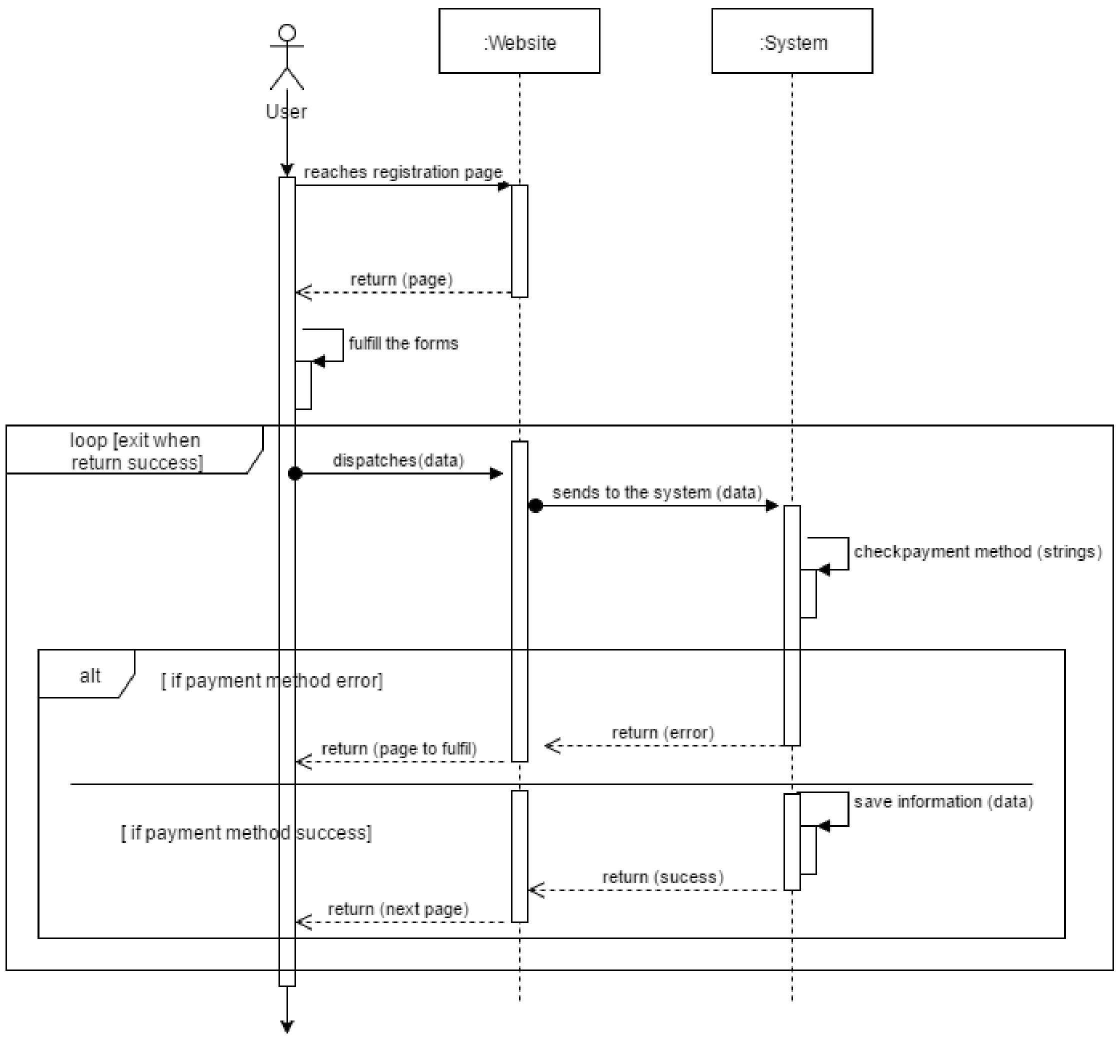
The diagram reveals the flow of activities when a user discover that his car has a problem or an incident has happened. Firstly, through the app he has the chance to know the number of the telephone exchange of the system that he can call in case of emergency. An operator will be ready to answer to all the doubts related to the situation and guide the user in solving the issue. As soon as the operator realizes that a problem occurred, he terminates the ride and set the car not available for other users. At this point he decides if a mechanic is needed for the situation and in case he calls him. Finally, the operator sends detailed information to the user about what to do and in case of an incident the rules to follow and where the useful documents (e.g. agreed motor accident statement) can be found.


operator
user
realize that the car has a problem
call the telephone exchange of the application
set the ride ended
set the car not available
is a mechanic needed ?
no
yes
call a mechanic
send instruction to the user
follow the instructions

3.7.3. Main sequences of the system

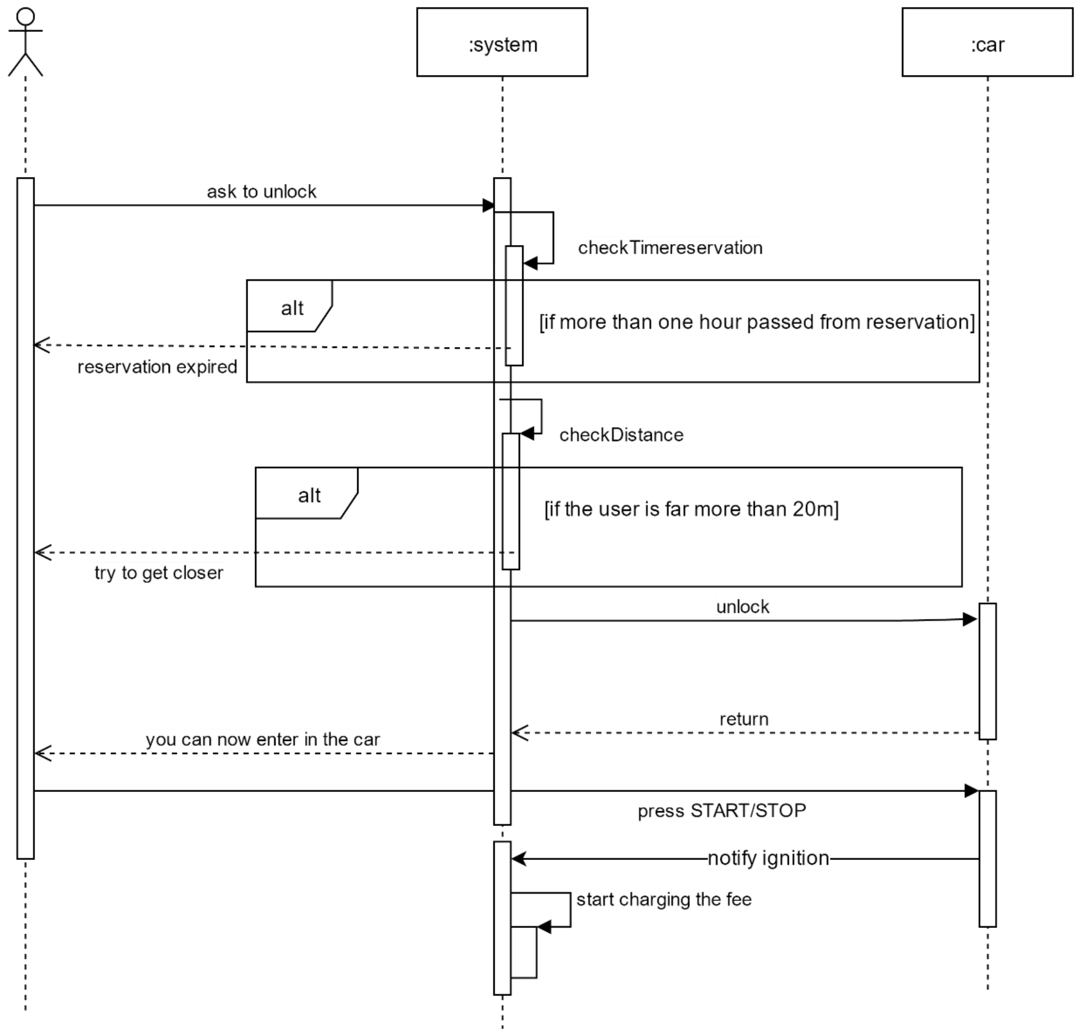
Sign Up Sequence Diagram

The following Sequence Diagram represents the Sign Up functionality. First of all, the unregistered user reaches the web-page. After the fulfilment of all the forms, the unregistered user sends the page to the website. The website turns in the data to the system, and he will check the payment method. As we can see there is a loop that holds until the unregistered user fill properly all the forms. When the system has successfully confirmed the payment method, the new user is able to reach the next web page.


User
:Website
:System
reaches registration page
return (page)
fulfill the forms
loop [exit when return success]
dispatches(data)
sends to the system (data)
checkpayment method (strings)
alt
[if payment method error]
return (error)
return (page to fulfil)
save information (data)
[if payment method success]
return (sucess)
return (next page)

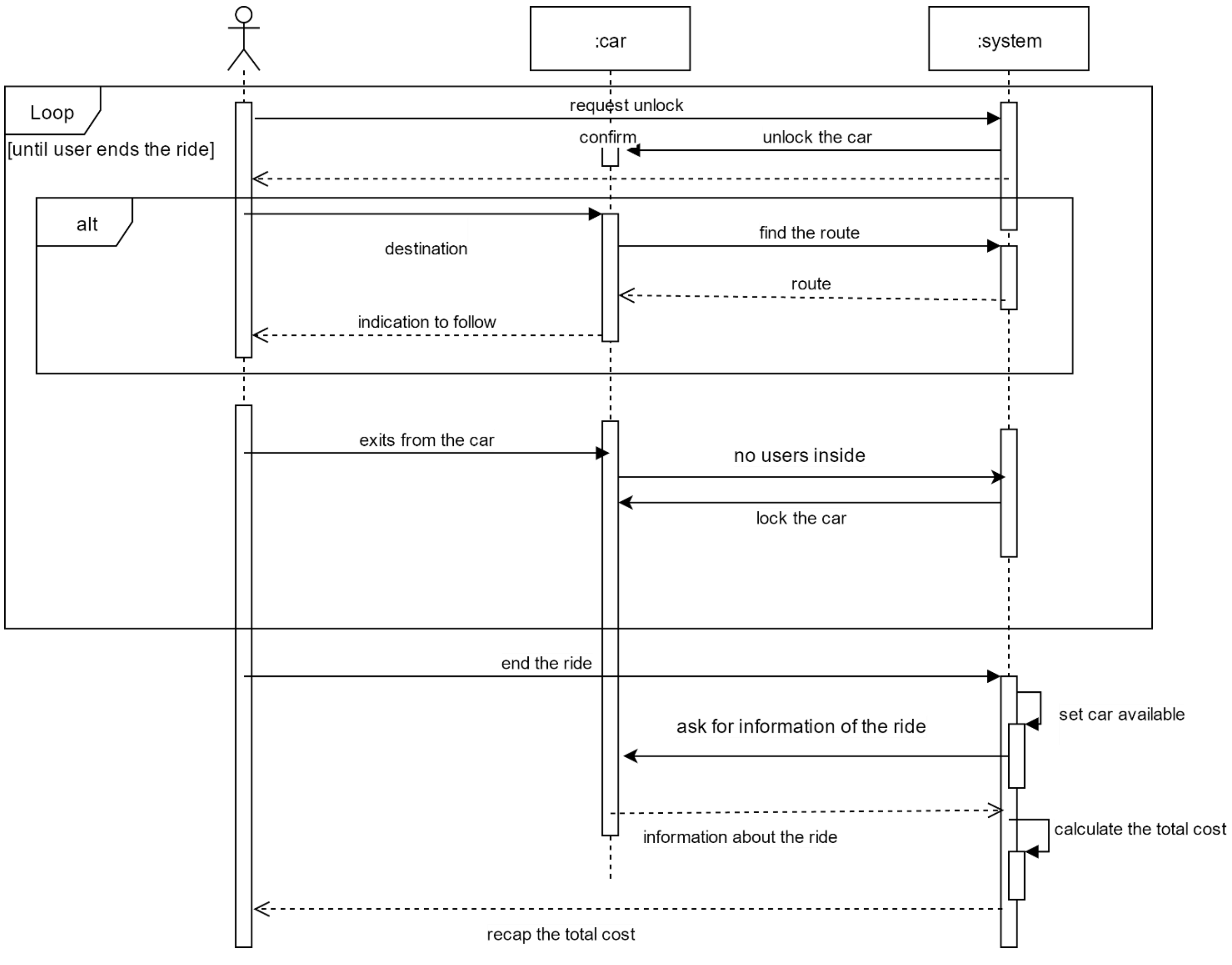
Begin the ride Sequence Diagram

Once the user has reached the car asks to the system, through the app, to unlock the car. The system checks the distance between the user (position provided by the GPS on his device) and the car itself. If the distance is under 20m and the request arrives within one hour from the reservation, the car is unlocked and the user can get on. By simply pressing the start/stop button present on the car the engine ignites and the car sends to the system the actual time in order to calculate the charge of the ride.



Complete Ride Sequence Diagram

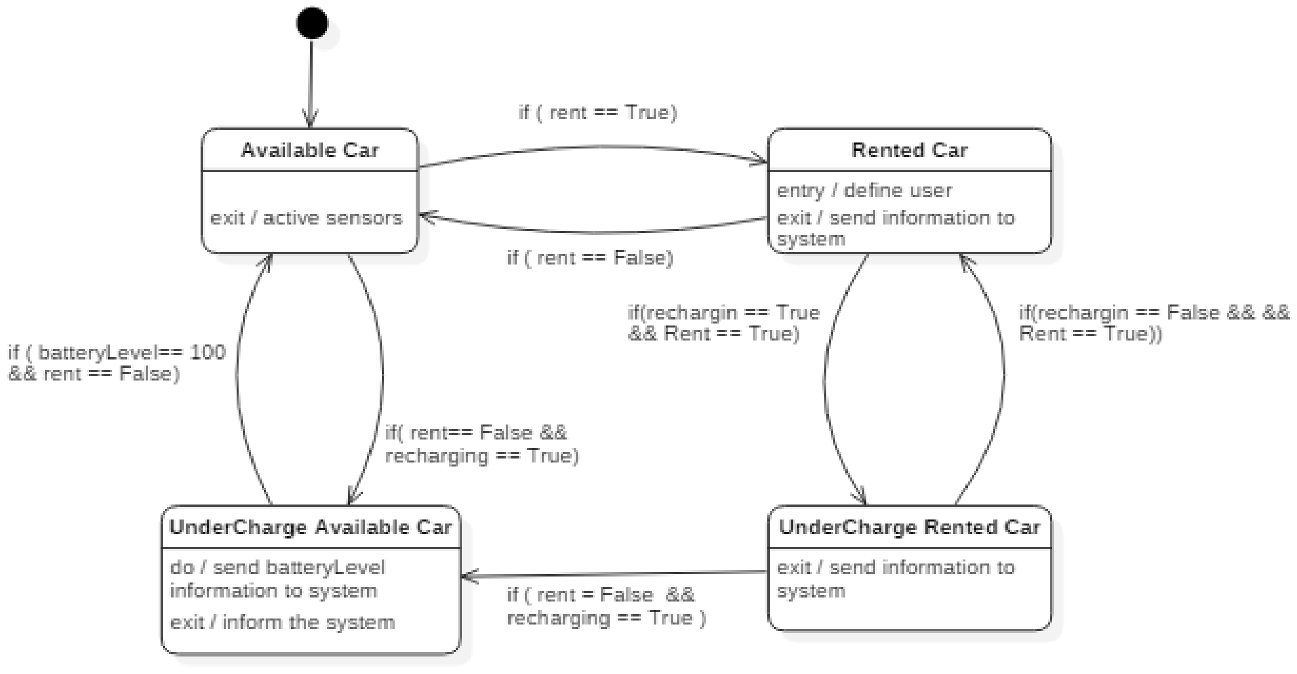
The diagram shows the interaction between the entities of our system during a ride, from the moment the car is unlocked to the end of the ride. the user, through the app, ask the system to unlock the car that forward the request to the car. Each user can take advantage, if desired, of the money saving option offered by system. Indeed, the user insert the final destination on the GPS system present on the car that asked to the system the most suitable route to follow in order to maintain a balanced distribution of the cars in the city. After the ride is completed the user exit from the car that is locked. At this point, the user can decide to end the ride by pressing a button on the app that notify the system to set the car available again and calculate the final cost. On the contrary, the user can decide not to end the ride so when is close enough can ask the system to unlock the car again and resume his ride. However, the time spent out of the car by the user is calculated normally.



3.7.4. State Chart Diagram

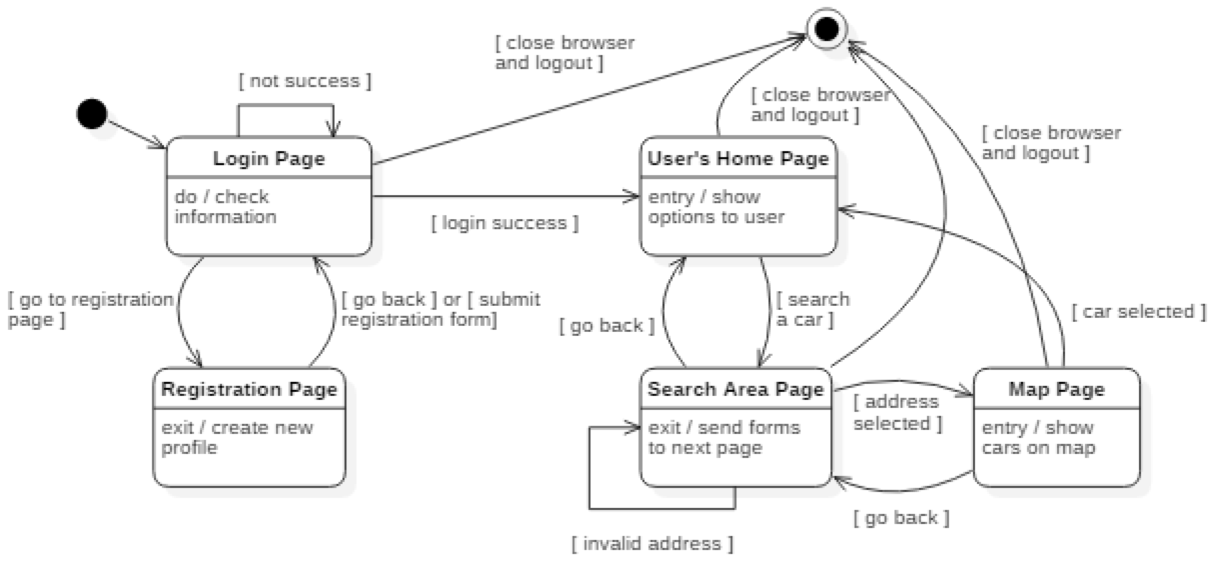
Car States

The following diagram would give the information related the possible states in which the car could be. First of all, the car is available and any user is able to rent it. After the rental the car changes its state. In this new state the user can recharge the car or end the rental. During the rental any recharge activities inside a special safe area is free of charge, but outside the special safe area the user needs to pay at their own expense. As we can see from the diagram the car can be recharged also during the available state.


Available Car
exit / active sensors
Rented Car
entry / define user
exit / send information to system
if (rent == True)
if (rent == False)
if (batteryLevel== 100 && rent == False)
if(rent== False && recharging == True)
if(rechargin == True && Rent == True)
if(rechargin == False && && Rent == True))
UnderCharge Available Car
do / send batteryLevel information to system
exit / inform the system
UnderCharge Rented Car
exit / send information to system
if (rent = False && recharging == True)

Browser Application States

The following diagram would give the information related the possible states of the web application of PowerEnJoy service. The diagram begins when one unregistered user reaches the website. In the website the unregistered/registered user can find all the information related the service but also a login form. If the user hasn't a profile, he can create one in the "Registration page". After the login several options, related to the service, are shown to the user. The user can select "search a car" from the "User's Home Page". In the "Search Area page" the user can decide to use his position or an address in order to search a car. After that a "Map page", which contains all the car inside a specified radius.



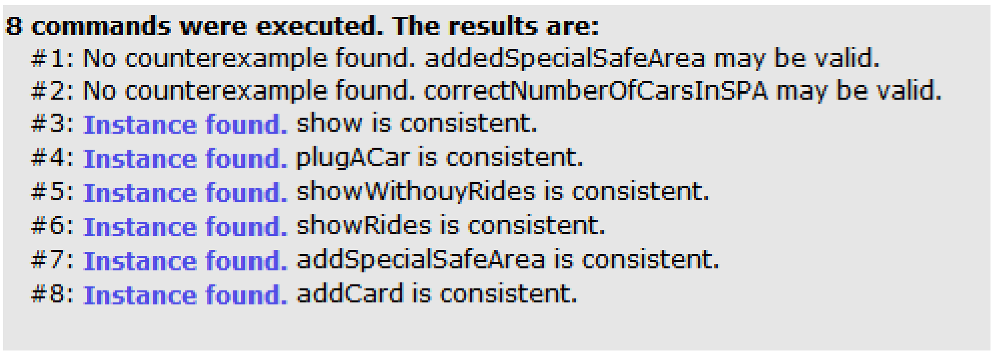
4. Model Coherence Analysis

4.1. Alloy Code

```
module PowerEnJoy



sig Validity{
    value: one String
}{
    all v: value | (v="VALID" or v="EXPIRED")
}

sig Card{
    validity: one Validity
}

sig User{
    drivingLicenceValidity: one Validity,
    cardSet: set Card
}{
    #cardSet>0
}

sig SafeArea{
    contains: set SpecialSafeArea
}

sig SpecialSafeArea{
    numberOfPlugs: one Int
}{
    all n: numberOfPlugs | n>0
}

sig Car{
    parked: lone SafeArea,
    plugged: lone {SpecialSafeArea+ExternalChargingStation}
}

sig Reservation{
    user: one User,
    car: one Car
}

sig Ride{
    reservation: one Reservation,
    statusOfRide: one String
}{
    all v: statusOfRide | (v="IN USE" or v="ONLY RESERVED")
}
```

```
sig Employee{
    workingOnCars: set Car
}

one sig ExternalChargingStation{}



fun cardsThatUseThatValidity[v: Validity]: set Card{
    {c: Card | v=c.validity}
}

fun usersThatUseThatValidity[v: Validity]: set User{
    {u: User | v=u.drivingLicenceValidity}
}

fact noUselessValidities{
no v: Validity | #cardsThatUseThatValidity[v]=0 and #usersThatUseThatValidity[v]=0
}

fun employeesWorkingOnThatCar[c: Car]: set Employee{
    {e: Employee | c in e.workingOnCars}
}

fun reservationWithThatCar[c: Car]: one Reservation{
    {r: Reservation | r.car=c}
}

fact freeCarMustAlwayBeParked{
all c: Car | (#employeesWorkingOnThatCar[c]=0 and #reservationWithThatCar[c]=0)
implies (#c.parked=1)
}

fact carThatAreParkedAndInChargeCantBeInAReservation{
    all c: Car | no r: Reservation | #c.parked=1 and #c.plugged=1 and c=r.car
}

fact oneSPAInOnlyOneSA{
    all spa: SpecialSafeArea | one sa: SafeArea | spa in sa.contains
}

fact expirationOfCards{
all r: Reservation, u: User | some c: Card | (r.user=u) implies (c in u.cardSet and
c.validity.value="VALID")
}

fact expirationOfDrivingLicense{
    no re: Reservation | re.user.drivingLicenceValidity.value="EXPIRED"
}

fact oneRideOneReservation{
    no disj r1, r2: Ride | r1.reservation=r2.reservation
}
```

```
fact differentCardsForEachUser{
    all c: Card | one u: User | c in u.cardSet
}

fact eachReservationMustHaveADifferentCarAndUser{
    (no disj r1, r2: Reservation | r1.user=r2.user)
    (no disj r1, r2: Reservation | r1.car=r2.car)
}

fact carStatusCondition{
(all c: Car, r: Ride | (c=r.reservation.car and r.statusOfRide="IN USE") implies
(#c.parked=0))
    (all r: Ride | r.statusOfRide="ONLY RESERVED" implies
#r.reservation.car.parked=1)
}

fact alwaysARideForAReservation{
    all re: Reservation | one ri: Ride | re=ri.reservation
}

fact workingCarCantBeReserved{
all e: Employee, r: Reservation | no c: Car | (c in e.workingOnCars) and (c=r.car)
}

fact occupationCar{
    all c: Car, e: Employee | (c in e.workingOnCars) implies (#c.parked=0)
}

fact parkedAndPluggedOnlyIfInSafeArea{
(all c: Car, sa: SafeArea, spa: SpecialSafeArea | (c.parked=sa and c.plugged=spa)
implies (spa in sa.contains))
    (all c: Car, ecs: ExternalChargingStation | (c.plugged=ecs) implies (#c.parked=0))
}

fun carsParkedInSPA[spa: SpecialSafeArea]: set Car{
    {c: Car | c.plugged=spa}
}

fact checkNumberOfCarPluggedForEachSPA{
    all spa: SpecialSafeArea | #carsParkedInSPA[spa] <= spa.numberOfPlugs
}

fact spaInSa{
    all c: Car | one sa: SafeArea, spa: SpecialSafeArea | ((c.plugged=spa) and
     (c.parked=sa)) implies (spa in sa.contains)
}



assert addedSpecialSafeArea{
     all spa: SpecialSafeArea, sa: SafeArea, sa': SafeArea | (spa not in sa.contains)
     and  addSpecialSafeArea[sa, sa', spa] implies (spa in sa'.contains)
}

check addedSpecialSafeArea for 3

assert correctNumberOfCarsInSPA{
    all spa:SpecialSafeArea | #carsParkedInSPA[spa] <= spa.numberOfPlugs
}

check correctNumberOfCarsInSPA for 3



pred plugACar(c: Car, spa: SpecialSafeArea){
    c.plugged=spa
}

pred addCard(u: User, c: Card){
    u.cardSet=u.cardSet + c
}

pred addSpecialSafeArea(sa: SafeArea, sa': SafeArea, spa: SpecialSafeArea){
    sa'.contains = sa.contains + spa
}

pred showWithoutRides{
    #Ride=0
}

pred showRides{
    #Ride>1
    #Employee=0
}

pred show{
}

run show for 4
run plugACar for 3
run showWithoutRides for 2
run showRides for 2
run addSpecialSafeArea for 3
run addCard for 3
```

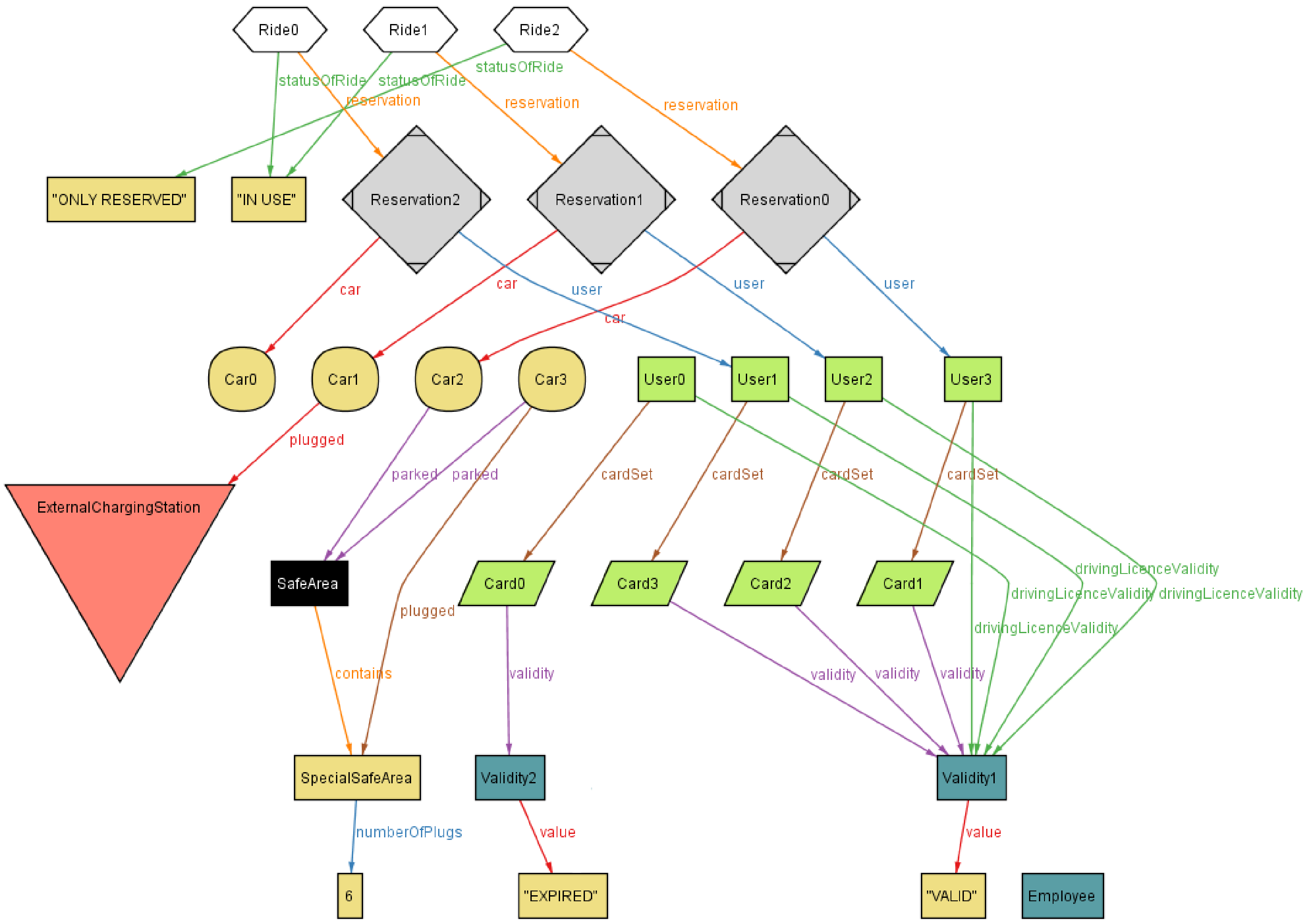


```
8 commands were executed. The results are:
  #1: No counterexample found. addedSpecialSafeArea may be valid.
  #2: No counterexample found. correctNumberOfCarsInSPA may be valid.
  #3: Instance found. show is consistent.
  #4: Instance found. plugACar is consistent.
  #5: Instance found. showWithouyRides is consistent.
  #6: Instance found. showRides is consistent.
  #7: Instance found. addSpecialSafeArea is consistent.
  #8: Instance found. addCard is consistent.
```

4.2. Generated World

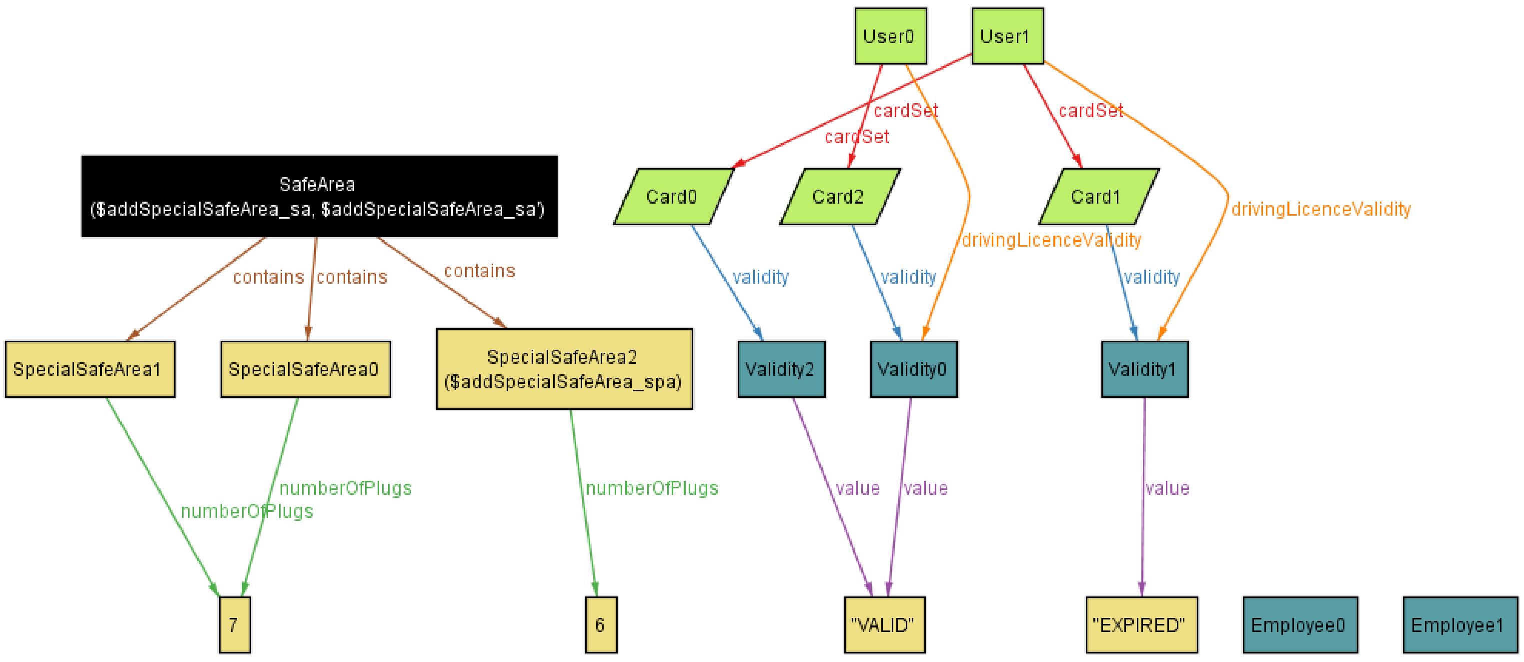
Alloy provides a snapshot of a possible situation of the reality considered. In the representations are included all the current active rides, while the already terminates ones are stored in a database in order to send the monthly invoice to users. In this reality we have also include the ExternalChargingStation in order to simulate this possible dynamic that may happen during a ride.

World: general world



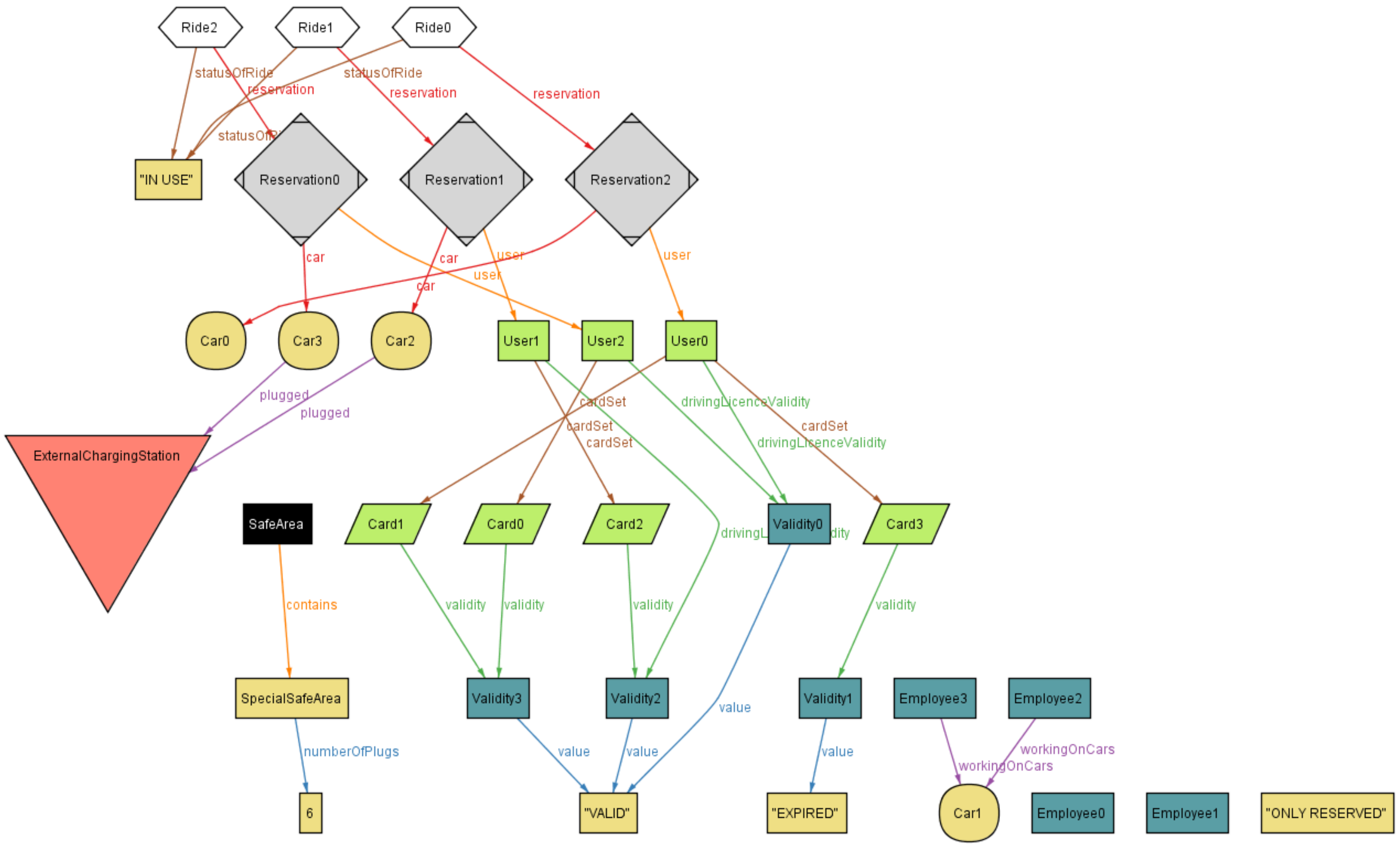
The diagram illustrates a snapshot of the reality where there are three different active rides. The first one, RIDE0, is currently "IN USE" and not in charge so the car is not parked anywhere. Then, RIDE1, where the car is "IN USE" but at the same time is charging in an ExternalChargingStation. In this case, the user cannot park the car there because all the ExternalChargingStation are considered outside from the safe areas. Lastly, RIDE2, where the car is parked inside a safe area and therefore not “IN USE” but “ONLY RESERVED”. In the diagram we can notice that Car3 is not under maintenance and is not part of any reservation so it must be parked and possibly plugged in a special safe area.

World: Add special safe area



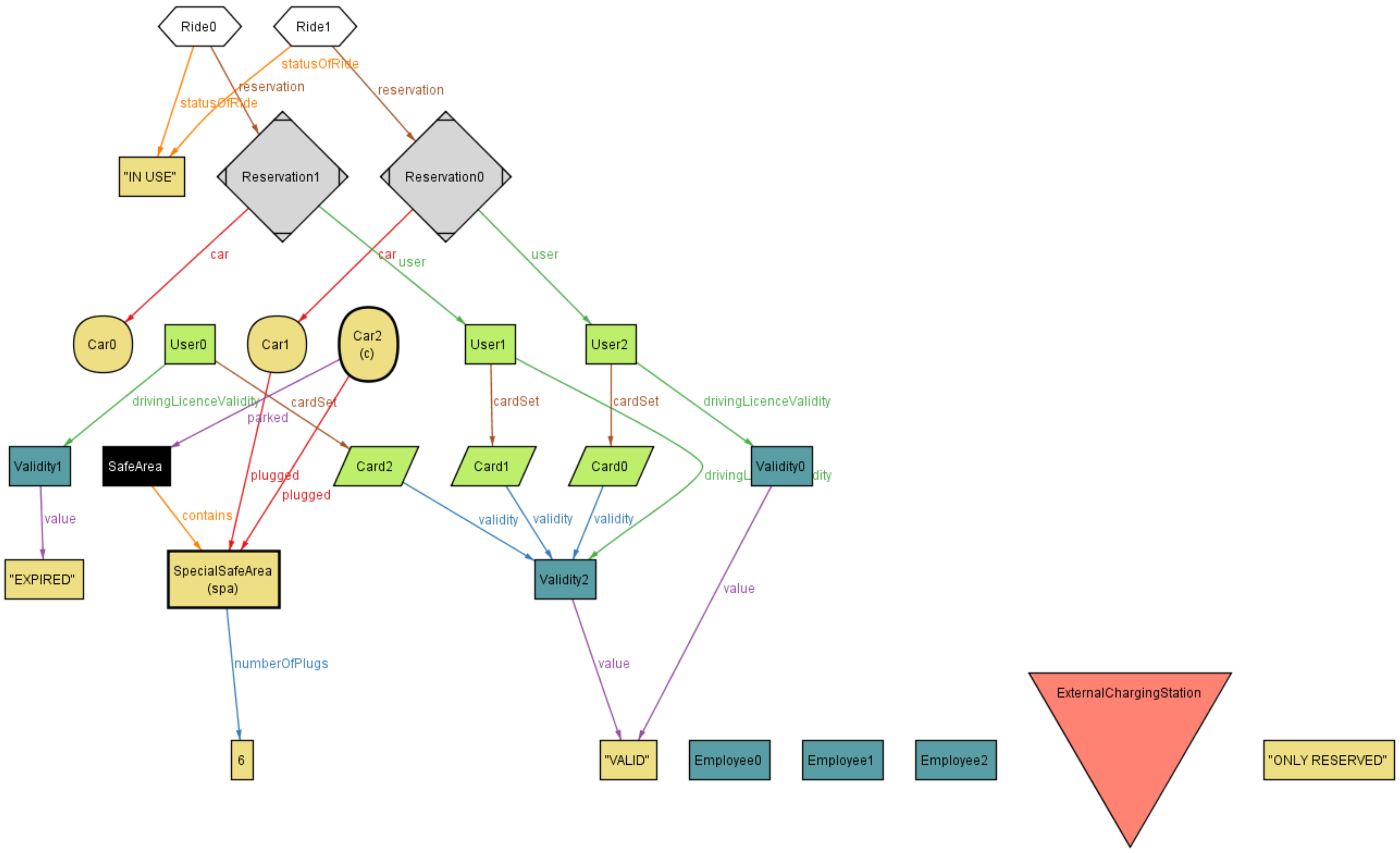
The diagram shows the dynamic evolution of our reality, when a special safe area is added to a safe area. In this case SpecialSafeArea2 is added and now SafeArea contains three different special safe areas. This diagram is focused only on the representation of the structure of SafeAreas and users and as consequence do not include the other objects.

World: Employees' roles



This diagram well illustrates the employees' roles that are working on a car. In this case Car1 is under maintenance and therefore cannot be included in any ride or parked because unavailable. In addition, there are other two employees that are waiting for calls from users or problem noticed by the system. Another important characteristic present is the possibility of having more cards (payment methods) associated with a user. In this case User0, has an expired card in his card set but can complete a reservation because has an additional valid card.

World: Plug a car



This last diagram shows how a car can be parked in a safe area and plugged in a special safe area. The special safe area must be contained in the same safe area and in addition the number of cars connected must be below the maximum number of plugs.

5. Tools

- Microsoft Word 2015
- StarUML
- www.draw.io
- Balsamiq Mockups3
- Alloy4.2

5.1. Hours of work

Lorenzo Frigerio:

- 18/10/16: 3h
- 23/10/16: 4h
- 25/10/16: 2h
- 1/11/16: 3h
- 2/11/16: 4h
- 6/11/16: 4h
- 8/11/16: 3h
- 10/11/16: 3h
- 12/11/16: 2h
- 13/11/16: 3h

Martina Perfetto:

- 18/10/16: 3h
- 23/10/16: 4h
- 25/10/16: 2h
- 28/10/16: 3h
- 29/11/16: 2h
- 1/11/16: 3h
- 2/11/16: 2h
- 6/11/16: 4h
- 8/11/16: 3h
- 10/11/16: 3h
- 13/11/16: 3h

Ivan Vigorito:

- 18/10/16: 3h
- 23/10/16: 4h
- 25/10/16: 2h
- 1/11/16: 3h
- 2/11/16: 4h
- 6/11/16: 4h
- 8/11/16: 3h
- 10/11/16: 3h
- 12/11/16: 3h
- 13/11/16: 3h

5.2. ChangeLog

- Class diagram cardinalities fixed (16/11/2016)
- Use case fixed (8/12/2016)

- Added world and machines (03/02/2017)