Software Engineering 2: PowerEnJoy

Requirements Analysis and Specification Document

Nardo Loris, Osio Alberto

Politecnico di Milano

Table of contents

1 Introduction

1.1 Purpose

This document is the Requirement Analysis and Specification Document for the PowerEnjoy system. Its aim is to completely describe the system, its components, functional and non-functional requirements, constraints, and relationships with the external world, and to provide typical use cases and scenarios for all the users involved.

1.2 Scope

The system is an electrical car sharing management system. Its main goal is to provide easy access to the service for the end user and to incetivize virtuous behaviours of the same users. The system consists in a web application addressed to two types of users:

- PowerEnjoy employee (Employee)
- Generic people (User)

The system allows non registered users to sign up and registered users to login with their credentials. The system must know the location of each car and logged in user. Registered users are allowed to reserve a single car in each geographic area at most one hour before picking it up. When a user is near a car he has reserved, if reservation has not expired, he asks for unlocking it, gets in and ignites the engine, starting a ride. A ride ends when the car is parked in a safe area and all the people inside it have got off. If reservation expires, user will no longer be able to pick the reserved car. Moreover, user will be charged with a fee.

The system allows the employee to manage safe areas and geographical regions to improve the overall offered service. Employees are also allowed to monitor the status of the car, so that when a car has a low battery charge an employee can move the car to a recharging station and plug the car or go and recharge the car on site.

1.3 Glossary

- **System**: the system to be developed for PowerEnJoy
- **User**: a generic person interacting with the system
- **Employee**: a person who works for PowerEnJoy
- **Car**: an electric vehicle owned by the company
- **Bill**: an amount of money a user has to pay. It is related to only a single ride. Each bill includes the date of issue.

- **Pending bill**: a bill that the user has not paid yet

- **Paid bill**: a bill that the user has already paid for

- **Area**: a space delimited by a polygonal line whose vertices are a set of geographical coordinates

- **Geographical coordinates**: a tuple of latitute and longitude describing a location on Earth

- **Geographical region**: an area where a user can reserve at most one car. They do not overlap

- **Safe area**: an area where it is possible to park a car and optionally to recharge it, in order to make it available for another user

- **Safe parking area**: a special safe area where it is not possible to recharge the car

- **Recharging station area**: a special safe area where it is possible to plug the car for recharging its battery

- **Registered user**: a user who has completed the sign up process

- **Logged user**: a user who has completed the log in process and has not yet started the log out process

- **Banned user**: a registered user who cannot reserve a car until all his pending bills are estinguished

- **Reservor user**: the user who has made a reservation for the specific car. A user is considered the reservor user of a car until the reservation expires or the user is charged with the bill

- **Available car**: a locked car for which no reservation exists

- **Reserved car**: a locked car for which it exists a user who has reserved it

- **Becoming available car**: an unlocked car is said to be "becoming available" as soon as all the passengers and the driver of this car exits the car, the doors of the car are closed and it is parked in a safe area

- **In maintenance car**: a locked car is said to be "in maintenance" as soon as its battery level is below 20%, the car cannot be reserved

- **In use car**: an unlocked car which is not becoming available

- **GPS**: A system capable of providing the location of a receiver device with a good precision (5 meters)

- **Overlapping areas**: Two areas are said to be overlapping if there exists at least one geographical coordinates which is contained inside the two areas

- **Expiration of a car reservation**: when a reservation expires, the car becomes available again, the reservor user loses his reservation and he is charged a fee of 1€

- **Percentage delta**: a discount or a raise based on percentage

- **Applying a raise** or **a discount**: The operation of increasing or reducing the amount of a bill for a specific reason. The amount is computed just before the system charges the user of a bill, and then all those amounts (each one related to a specific reason) are algebraically added to the same bill.

1.4 Goals

- [G1] Users must be able to sign up to the system entering their name, birth date, address, credit card information and a valid driving license number, in addition to username and password

- [G2] Only registered users and employee must be able to log in to the system
- [G3] Logged users must be able to find the location and the level of the battery of available cars within a user defined distance from his current location
- [G4] Logged users must be able to find the location and the level of the battery of available cars within a user defined distance from a user defined address
- [G5] Logged users must be able to reserve for each geographical region at most one car among the available ones
- [G6] The reservation of a car expires after one hour
- [G7] Only the reservor user must be able to unlock the car he has a reservation for
- [G8] When a user ignites the engine of a car, the system starts computing the bill
- [G9] As long as a car remains in use, the amount of the bill of the reservor user of that car is increased for a per-car defined amount of money per minute
- [G10] Car drivers are notified of the current bill through a screen on the car
- [G11] When a car is becoming available, then the bill can be charged to the reservor user
- [G12] The system must incentivize the virtuous behaviors of the users
- [G13] After being used by its reservor user, a car will be available for a new reservation after the system has charged that user with the bill corresponding to his ride
- [G14] A user that cannot pay his bill is banned until the bill is estinguished
- [G15] Logged users must be able to find all their reservation that are still valid
- [G16] Logged employees are able to view all the geographical regions
- [G17] Logged employees are able to split geographical regions
- [G18] Logged employees are able to merge geographical regions
- [G19] Logged employees are able to view all the safe areas
- [G20] Logged employees are able to remove a safe area
- [G21] Logged employees are able to add a new safe area
- [G22] Logged employees are able to update the information of a safe area
- [G23] Logged employees can view all the cars which are in mantainace and are not plugged into a power grid

1.5 References

This document refers to the rules of the Software Engineering 2 project and to the RASD Assignment.

1.6 Overview

This document is structured in three parts:

- **Chapter 1: Introduction** It provides an overall description of the system scope and purpose, with some information about this document
- **Chapter 2: Overall description** It provides a general perspective over the main system features, constraints and assumptions about the users and the environment
- **Chapter 3: Specific requirements** It focuses in the functional and non-functional requirements.

2 Overall description

2.1 Product perspective

2.1.1 User interfaces

The system will present itself as a web application, the interface will be intuitive and responsive.

2.1.2 Hardware interfaces

The system should use the following hardware interfaces:

- GPS interface to obtain the position of car and logged users
- Car monitoring system to obtain the charge level of the battery, whether the engine is ignited, the number of passengers, whether all the doors are closed.

2.1.3 Software interfaces

- The system will run on any web server that supports Java Enterprise Edition and stores its data in a RDBMS.

2.2 Product functions

The system allows:

- The users to:
  - Sign up
  - Login
  - Search nearby available cars
  - Search available cars nearby a specific address
  - View all the non-expired reservations
  - Make a reservation

- Unlock a car previously reserved
- The employee to:
    - Login
    - View the status of all the cars
    - Manage geographical regions
    - Manage safe areas

2.3 User characteristics

The system has two classes of users: PowerEnJoy employees and end user of shared cars. All of them have access to the Internet, and can use a browser.

2.4 Constraints

2.4.1 Regulatory policies

The users are responsable of the information provided and for any unlawful behaviour. The system must ask the user permission for acquiring, processing and storing personal data.

2.4.2 Hardware limitations

The system has to run under the following worst-case conditions:

- 1 Mb/s Internet connection
- Viewport size of 320x480 pixels

2.4.3 Interfaces to other applications

- External system for driving licence validation

    The interaction with external system for driving licence validation is carried out over HTTPS requests

- External system for address translation

    The interaction with external system for address translation is carried out over HTTPS protocol

- External system for payments processing

The interaction with external system for payments processing is carried out over a SOAP API offered by the same service

2.4.4 Reliability requirements

The system must have a minimum availability of 99%.

2.4.5 Criticality of the application

The system is not used in life-critical applications.

2.4.6 Safety and security considerations

- The user's payment information and driving licence number and their location must be kept private.
- The position of in use cars must be visible only to the employee.

2.5 Assumptions and dependencies

- [D1] Driving license number can be checked by an external service
- [D2] There exists an external service for processing payments
- [D3] An external service is available to get the geographical coordinates of an address
- [D4] Each car has a display capable of showing the current bill amount
- [D5] Car can communicate with the system via mobile internet connection
- [D6] Car position is detected using GPS
- [D7] The level of charge of the battery is detected by the system
- [D8] The number of people in the car is detected by the system
- [D9] The system knows whether a car has been plugged for recharge or not
- [D10] The system knows whether the engine is ignited or not
- [D11] If the system unlocks a car, anyone can enter the car
- [D12] If the system locks a car, nobody can enter the car
- [D13] The user position is detected using GPS
- [D14] A user interacting with the system has an internet connection
- [D15] Geographical regions do not overlap
- [D16] Safe areas do not overlap
- [D17] Each employee has its own username and password (different from those used to reserve cars)
- [D18] The system is able to detect whether the doors of a car are opened or closed

- [D19] Car plates are unique

- [D20] Driving licence are unique

- [D21] Cars are managed via an external system, that the company already owns

2.6 Future extensions

The system will be implemented foreseeing the possibility of further extensions. An incomplete list of possible extensions is below:

- Allow a user to modify the data sent during sign up

- Allow a user to unregister from the system

- Programmatically assign in maintenance cars to employees

- Provide the users with a money saving option, letting them input their destination and providing informations about where to leave the car, as to provide a better distribution of cars.

- Keep track of virtuous behaviours of the users, in order to provide the most virtuous users with further discounts.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

The user interfaces must satisfy the following constraints:

- Web pages must comply with the W3C and ECMA International standards. In particular to the HTML5, CSS and ECMA-262 specification.

- The first screen must ask the user to login or to sign up in order to begin operations.

- There must be a menu which lists any functionality available for the user, in order to make the usage of the application simpler

- The menu must be visible in any page

- The user interface must provide breadcrumbs navigation

- The compilation of input fields must be done with suitable controls to simplify user's interactions

- Web pages must be responsive

3.1.2 Hardware interfaces

The system should use the following hardware interfaces:

- GPS interface to obtain the position of car and logged users
- Car monitoring system to obtain the charge level of the battery, whether the engine is ignited, the number of passengers, whether all the doors are closed

3.1.3 Software interfaces

The system requires:

- MySQL 5.7
- Java EE7

3.1.4 Communications interfaces

Users and employees communicate with the system via HTTPS connection.

3.2 System features

3.2.1 Typical usage scenarios
Scenario 1

Alice and Bob live in Milan. One afternoon Bob wants to go and meet Alice. Since Bob's car is broken, he decides to take advange of PowerEnJoy car sharing. Then he registers to the system and reserves a car. The reserved car appears in "my reservations" section on the application. He reaches the car, unlocks and ignites it. When he reaches Alice's house, Bob looks for a car park to leave the car. Once he gets out of the car, the system locks it automatically. Bob is happy to discover that, having left the car with less than 50% battery empty, the system has applied a discount on his fee.

Scenario 2

Alice and Bob live in Milan. Bob is very interested in modern art, and asks Alice to go with him to visit an exhibition about Escher. Alice knows that Eve is interested in the exhibition too, and asks her to come with them. The three friends don't want to drive their own car in the center of Milan, so, as Bob is registered to PowerEnJoy car sharing system, they decide to use a shared car of that company. Bob reserves a car in the nearest parking station, and then goes to lunch with Alice and Eve. After lunch, the friends reach the car and drive to the museum. As Bob has brought with him other two people, he is applied a discount on the fee.

Scenario 3

Mallory lives in Milan. Mallory's car is in maintenance, but she is in desperate need of a car because her favourite pop star will meet his fans at the Mediolanum Forum in Assago. She remembers about Bob talking about a very nice car sharing system, so she browses the web and founds PowerEnJoy website. She follows the registration instructions, then accesses the application, searches for the nearest car and reserves it immediately. After that, John calls Mallory to ask her to take part in a match of Mallory's favourite online game. Mallory thinks that she has plenty of time to end the match before meeting her favourite pop star, but the match goes longer and longer. At the end Mallory reaches in a hurry the parking station where the reserved car is located, but she is not able to find any car, as she spent more then one hour since reservation playing that game. Moreover, Mallory is charged of a fee for having reserved and not used a car.

Scenario 4

John invites Mallory to try a match at the nearest *Laser Game* arena. Since Mallory is registered to PowerEnJoy car sharing system, he proposes to care about reserving a car. 30 minutes before meeting John, Mallory searches for the nearest car and reserves it. Then he goes to the car park, takes up the car and drives first to John's and then to the arena. As the arena is very far from the nearest parking station, the two friends decide to park the car just outside it, though that is not a car park. The system does not stop charging Mallory util the two friends end the game, drive back home and the car is parked again in a safe area.

Scenario 5

Alice wants to go to the fashion outlet to buy the latest dress of her favourite fashion designer. As Alice is very attentive to the environment and only drives electric and shared cars, she is a affectionate user of PowerEnJoy car sharing. Alice opens the application, searches for the nearest car and reserves it. The reserved car appears in the section "my reservations" on the application. When she reaches the supermarket she parks the car in the nearest recharging station and plugs it, as she knows that doing so she will get a discount on last ride.

Scenario 6

Mallory lives in Milan. One evening Mallory wants to reach her friends to a new disco that has just opened outside Milan. As usual, one of her friends will care of bring them back as they will be completely drunk, and so she doesn't want to take her car because she would have to leave it at the disco. Mallory decides to take a car from PowerEnJoy car sharing system, and then she drives to the disco. Since the disco is quite far from Mallory's house, when she arrives the car as quite no more batery charge left. When she parks the car, she is charged an additional fee for having left the car far from the nearest recharging station.

Scenario 7

Jasper leavs quite far from Milan, and to reach his office he drives to the nearest underground station and then takes the underground train. As there is a PowerEnJoy recharging station just outside the underground station, Jasper choses that company, registers and is able to take a car. When he reaches the station he would like to park in the apposite area, but all plugs are already in use, and so he has to park in usual car parks. The system charges Jasper with an additional fee for not having plugged a quite completely discharged car.

Scenario 8

Professor Talmoto has to take part in an important conference in Florence. He chooses to reach Florence by train. When he is quite near Florence, he opens PowerEnJoy application and searches for a car near the station selecting it basing on left battery charge, in order to have it ready when he gets off the train. He takes the car and drives to the hotel, being pleased by the fact that the car notifies him the current charge on the display. When he reaches the hotel, he parks the car in the nearest parking station, the system locks it and charges the professor with the fee.

Scenario 9

Rose is a very affectionate user of PowerEnJoy car sharing, and she daily use shared cars provided by that company. She is also very inattentive, and never remembers to charge her prepaid card. One day, after having parked her car, she receives a message that tells her it was not possible to complete the payment for her last ride. She knows that she will not be able to reserve and use cars until she pays her last ride, so she runs to the nearest bank to charge her credit card back, then opens her profile on PowerEnJoy's website and pays the ride.

Scenario 10

Mark is an employee of PowerEnJoy. His job includes the responsibility to maintain the list of safe parking areas and recharging areas updated. In order to update the list of those areas, Mark has to connect to the management system with his employee credentials. Then he can open a page with the list of safe parking areas and add, remove or update parking areas. He can also open another page dedicated to recharging stations and update, insert or remove them as well.

Scenario 11

Jack is one of the managers of PowerEnJoy. Jack is in charge of tuning the service in order to best fit user needs. Jack can modify the geographical areas in which users can reserve only one car in order to alter the way users reserve cars, and so the availability of the same cars. To do that, Jack connects to the management system with his company credentials and opens a map which shows how a certain territory is divided in geographical areas, and from here he can modify the boundaries of each area, split an area in two subareas or merge several areas into a single area. When Jack applies the changes, new reservations will be affected, but not old ones.

3.2.2 Detailed use case description
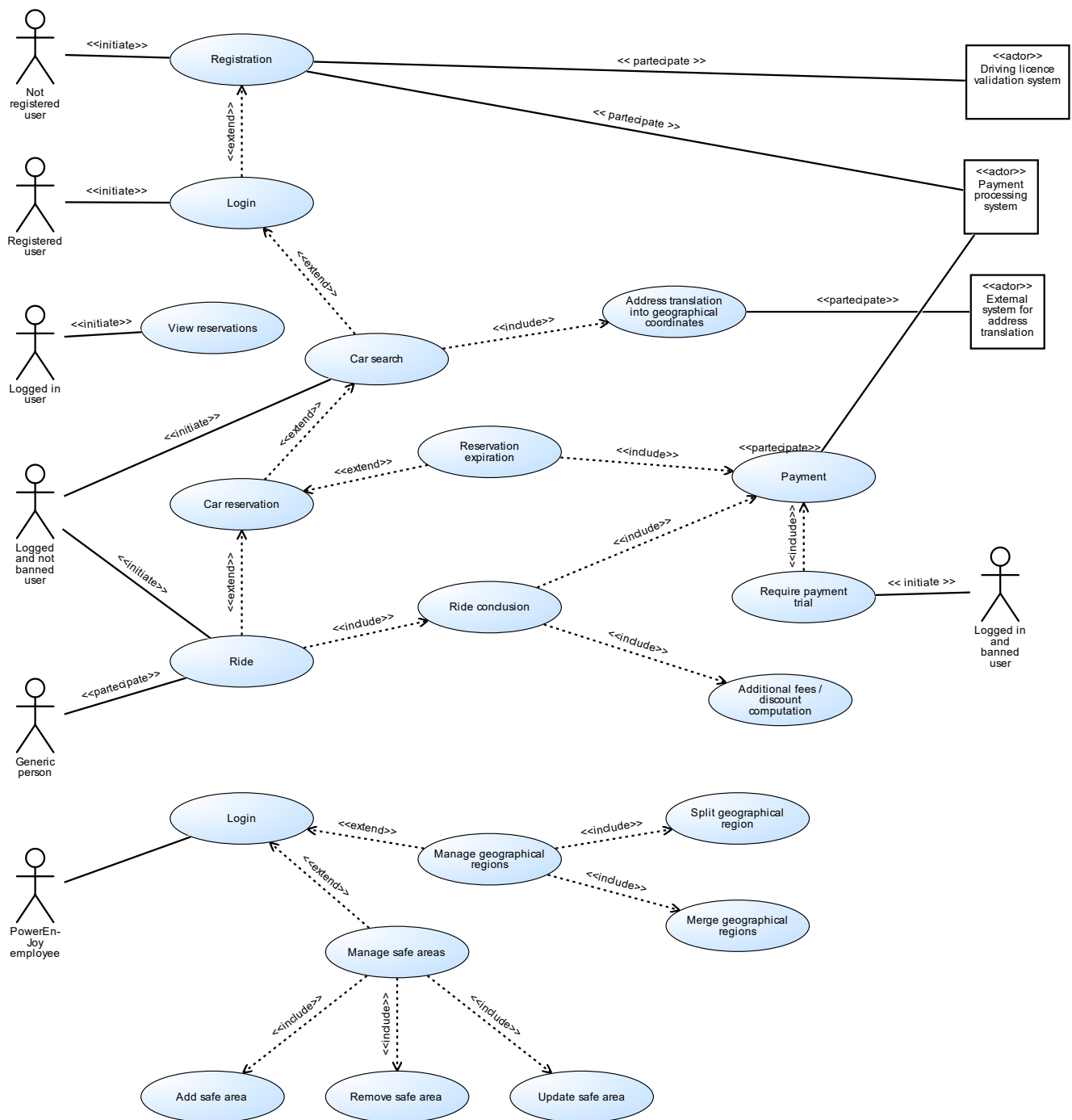
Registration

Actors

- User
- External system for driving licence validation
- External system for credit card parameters validation

Entry condition

- User is not registered to SYSTEM

Flow of events

- The user activates the *sign up* function on his terminal. SYSTEM responds by presenting him a form.
- The user fills the form, inserting his complete name, birth date driving licence number, credit card number and CVV, along with a username and a password. Once the form is completed, the user submits it.
- SYSTEM validates the inserted data, and in particular queries external systems for driving licence number and credit card informations validation. If all is correct, then SYSTEM stores the new user data

Exit conditions

- User is registered and can log in to the system
- User is not banned

Exceptions

- Information submitted to SYSTEM by the user is not complete or wrong. In this case, user is notified of the error(s) and registration procedure fails.
- External service for driving licence validation or external service for credit card information validation are unavailable. In this case user is notified of the unavailability of the registration service.

Login

Actors

- User

Entry condition

- The user is registered to the system

Flow of events

- The user activates the *sign in* function on his terminal. SYSTEM responds by presenting him a form.
- The user fills the form, inserting the username and password he had chosen at registration time, and then submits it.
- SYSTEM validates the credentials.

Exit condition

- User is logged in to the system and, if he's not banned, he can search for cars, else he can ask SYSTEM to try again to perform pending bill payment.

Exceptions

- Username or password are wrong. In this case, user is notified of the error and he is not logged in.

Car search

Actors

- User

Entry conditions

- User is logged in to the system
- User is not banned

Flow of events

- The user activates the *search a car* function on his terminal. SYSTEM responds by presenting him a form.
- User chooses whether to search for cars near his current position or near a target address, and fills the form entering the radius of the search and, if he has chosen to search by address, the target address, then he submits the form.
- If search by address was chosen, **address translation into geographical coordinates** use case is called
- SYSTEM queries the list of cars searching for available cars that fit user parameters (search center expressed by its geographical coordinates and search radius)
- SYSTEM responds to the user's terminal with a list of the available cars, each one charcterized by its unique identification number, its position (expressed by its geographical coordinates) and its battery level
- The user terminal builds a map in which car positions are highlighted with a pin reporting the car identification number and a list of all cars reporting, for each car, identification number, position (expressed by its geographical coordinates) and battery level

Exit condition

- The user has received a list of all cars that fit its search parameters

Exceptions

- If no car is found, the user is notified of the unavailability of cars

- If address cannot be translated into geographical coordinates, user is notified of the impossibility to perform the search

Address translation into geographical coordinates

Actors

- External system for address translation

Entry condition

- SYSTEM is processing an address-based search

Flow of events

- SYSTEM queries the external service with the address to translate
- The external service responds with the geographical coordinates of the address

Exit condition

- Address is translated into geographical coordinates

Exceptions

- External service is not available. In this case the caller is notified of the unavailability of the service.

Car reservation

Actors

- User

Entry conditions

- User is logged in
- User is not banned
- User has issued a car search and results have been presented on his terminal

Flow of events

- User activates the *reserve car* function on his terminal. A request carring as its payload the unique identification number of the selected car is sent to SYSTEM.
- SYSTEM checks that the car is actually available
- SYSTEM checks that the user has no reservation for other cars belonging to the region to which the selected car belongs
- SYSTEM reserves the target car for the user

- User is notified of the result of the operation

Exit conditions

- Car requested by the user is reserved
- The reservor user of the selected car is the user who issued the request

Exceptions

- If car is not available, SYSTEM notifies the user of the error. The user can then select another car.
- If the user has already a reserved car in the same geographical region, SYSTEM notifies the user of the error. The user can select another car.

Special requirements

- SYSTEM checks and car reservation must be performed as a unique atomic sequence.

View reservations

Actors

- User

Entry conditions

- User is logged in

Flow of events

- User activates the *my reservations* function on his terminal
- SYSTEM retrives the list of all still valid reservations (those neither used to actually take a car nor expired yet)
- SYSTEM responds to the user with the list of those actions

Exit conditions

- User can see a list of all his still valid reservations

Reservation expiration

Actors

- User
- External system for payment processing

Entry condition

- User has reserved a car

- Since reservation time, a time of one hour has elapsed

Flow of events

- SYSTEM cancels the reservation of the car for the user (the car is marked as available)

- SYSTEM charges the user with a fee of 1€

- SYSTEM invokes the ***payment*** use case in order to actually perform the payment

- External system for payments processing answers whether the payment attemp was successful or not

Exit condition

- User is able to reserve other cars

Exceptions

- If payment is not successful, user is marked as "banned" and he is notified of the error

- If external system is not available, user is marked as "banned" and he is notified of the error

Ride

Actors

- User

- Other people not necessarly registered to SYSTEM (referred to as passengers)

Initial condition

- User is logged in

- User is not banned

- User is the reservor user for a car

Flow of events

- User reaches the reserved car and activates the *unlock car* function on his terminal

- SYSTEM checks the position of the user, and, if the user is near the car, SYSTEM unlocks it.

- User enters the car. If there are some passengers, they enter the car too.

- User ignites the engine. The car signals to SYSTEM that the engine was ignited.

- SYSTEM creates an empty bill for the user.

Exit conditions

- The user can actually drive the car he had reserved.

- SYSTEM charges him with a per minute fee.

- Car communicates with SYSTEM in order to signal passengers entering or exiting the car, or that the ride is ended

Ride conclusion

Actors

- *(User is not directly involved, SYSTEM will charge him with the bill, but he actually does nothing)*

Entry condition

- The car reserved by the user has been parked in a safe area

Flow of events

- The car signals to SYSTEM that all passengers have got off
- SYSTEM locks the car
- SYSTEM closes the bill associated to the ride
- SYSTEM detects whether the car is parked in a safe parking area or in a recharging station. It the car is parked in a recharging station, SYSTEM waits for 5 minutes
- SYSTEM invokes the ***additional fees/discounts computation*** use case
- SYSTEM marks the car as "available"
- SYSTEM charges the bill to the user
- SYSTEM invokes the ***payment*** use case

Exit condition

- User is able to reserve other cars
- The car is available for another reservation

Exceptions

- If the payment is not successful (that is, it is rejected, for example because user has no money on the credit card), user is marked as "banned" and he is notified of the error
- If the payment could not be performed (for example, because the external system is not available), user is marked as "banned" and he is notified of the error

Additional fees/discounts computation

Actors

No other actor than SYSTEM is involved

Entry condition

- SYSTEM is processing the bill associated to a ride

Flow of events

- SYSTEM asks the car to send information about its battery level, its actual position and whether or not it is plugged to a power source
- The car sends required information to SYSTEM
- SYSTEM computes percentage fees or discounts on the final amount of the bill, accorring to these rules
    - -10% if for all the duration of the ride, SYSTEM detected that 3 or more people (including the driver) were con the car
    - -20% if residual battery level is greater or equal to 50% of full charge level
    - -30% if the car is actually plugged to a power source
    - +30% if battery level is less then 20% of full charge level
    - +30% if the car is parked more than 3km far from the nearest recharging station
- SYSTEM algebraically adds the computed amounts to the amount of the bill associated to the ride under consideration

Exit condition

- The user bill includes additional fees and discounts contribution

Payment

Actors

- External system for payments processing
- *(User is not directly involved, SYSTEM eventually changes his status, but he actually does nothing)*

Entry condition

- There is a pending bill in SYSTEM related to the user

Flow of events

- SYSTEM requests the external system to actually carry out the payment, sending in the request the amount of the bill and the credit card information of the user whom this bill belongs to
- The external system acknowledges the payment
- If the user bound to this bill was banned and now he has no more pending bills, the user is not anymore considered "banned"

Exit condition

- The bill is paid

Exceptions

- The payment cannot be carried out. In this case, the external system signals to SYSTEM the error.
- The external system is not available. In this case, SYSTEM stops the procedure and the bill is still marked as "pending"

Require payment trial

Actors

- User

Entry conditions

- User is logged in
- There exists at least one pending payment associated to the user (and thus the user is banned)

Flow of events

- User activates the *pay pending bills* function on his terminal. A request is sent to SYSTEM.
- SYSTEM searches all pending payments associated to the user
- For each found payment, SYSTEM activates the ***payment*** use case
- SYSTEM responds to the request acknowledging the completion of the payments

Exit conditions

- User is not anymore banned and can reserve cars

Exceptions

- If a payment fails, SYSTEM notifies the user of the error
- If a payment could not be performed, SYSTEM notifies the user of the error

Manage safe areas

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials

Flow of events

- The employee activates the *manage safe areas* function on his terminal. SYSTEM responds with a list of safe areas currently known (this list includes both safe parking areas and recharging stations)
- If the employee chooses to add a new safe area activating the *add safe area* function on his terminal, then the use case **add safe area** is activated
- If the employee chooses to remove a new safe area activating the *remove safe area* function on his terminal, then the use case **remove safe area** is activated
- If the employee chooses to update the parameters of a safe area activating the *update safe area* function on his terminal, then the use case **update safe area** is activated

Exit conditions

- The list of safe areas is updated

Add safe area

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials
- The employee has chosen to add a new safe area

Flow of events

- SYSTEM shows the employee a form
- The employee fills the form with the parameters of the new safe area (the type of the area, chosen between safe parking area and recharging station, the boundaries of the area and the number of plugs if he is inserting a recharging station), then submits the form
- SYSTEM adds the new safe area to the list of known areas

Exit conditions

- The new safe area is added to the list of safe areas

Exceptions

- If the new area overlaps with another safe area, than the request of adding it is rejected

Remove safe area

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials
- The employee has chosen to remove a safe area

Flow of events

- The employee chooses which safe area has to be deleted
- SYSTEM deletes the safe area from the list of known areas

Exit conditions

- The safe area is removed from the list

Update safe area

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials
- The employee has chosen to update a safe area

Flow of events

- The employee chooses which safe area has to be updated
- SYSTEM shows a form in which user can update safe area parameters, then submits the form
- SYSTEM updates the parameters of the selected safe area

Exit conditions

- The safe area parameters are updated

Exceptions

- If the safe area boundaries overlap with another parking area, than the request of updating it is rejected

Manage geographical regions

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials

Flow of events

- The employee activates the *manage geographical regions* function on his terminal
- SYSTEM sends to the employee's terminal a map showing all the geographical regions
- If the employee selects one region, he can decide to divide it into two regions or to merge it with another region. In the former case **split geographical region** use case is activated, in the latter **merge geographical regions** use case is activated

Exit conditions

- Geographical regions are updated
- New reservations will be affected by the update
- Reservations that are already inserted are not affected by the update

Split geographical region

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials
- The employee has selected a region from the ones inserted in the system
- The employee has chosen to split that region

Flow of events

- SYSTEM presents on the employee's terminal a map where selected geographical region is highlighted
- The employee draws a line across the region so that the geographical region is divided in two subregions
- The employee confirms his choice
- SYSTEM updates the set of geographical regions removing the old geographical region and adding the two subregions

Exit conditions

- The geographical region selected by the employee is now split into two subregions

Merge geographical regions

Actors

- PowerEnJoy employee

Entry conditions

- The employee is logged in with his company credentials
- The employee has selected a region from the ones inserted in the system
- The employee has chosen to merge that region with another region

Flow of events

- SYSTEM presents on the employee's terminal a map where geographical regions adjoining the selected one are highlighted
- The employee selects one of the highlighted geographical regions
- The employee confirms his choice
- SYSTEM updates the set of geographical regions removing the two selected geographical regions and adding one geographical region which is the result of the union of the two deleted geographical regions

Exit conditions

- The geographical regions selected by the employee are now merged into a single region

3.2.3 Domain models

Domain class diagram
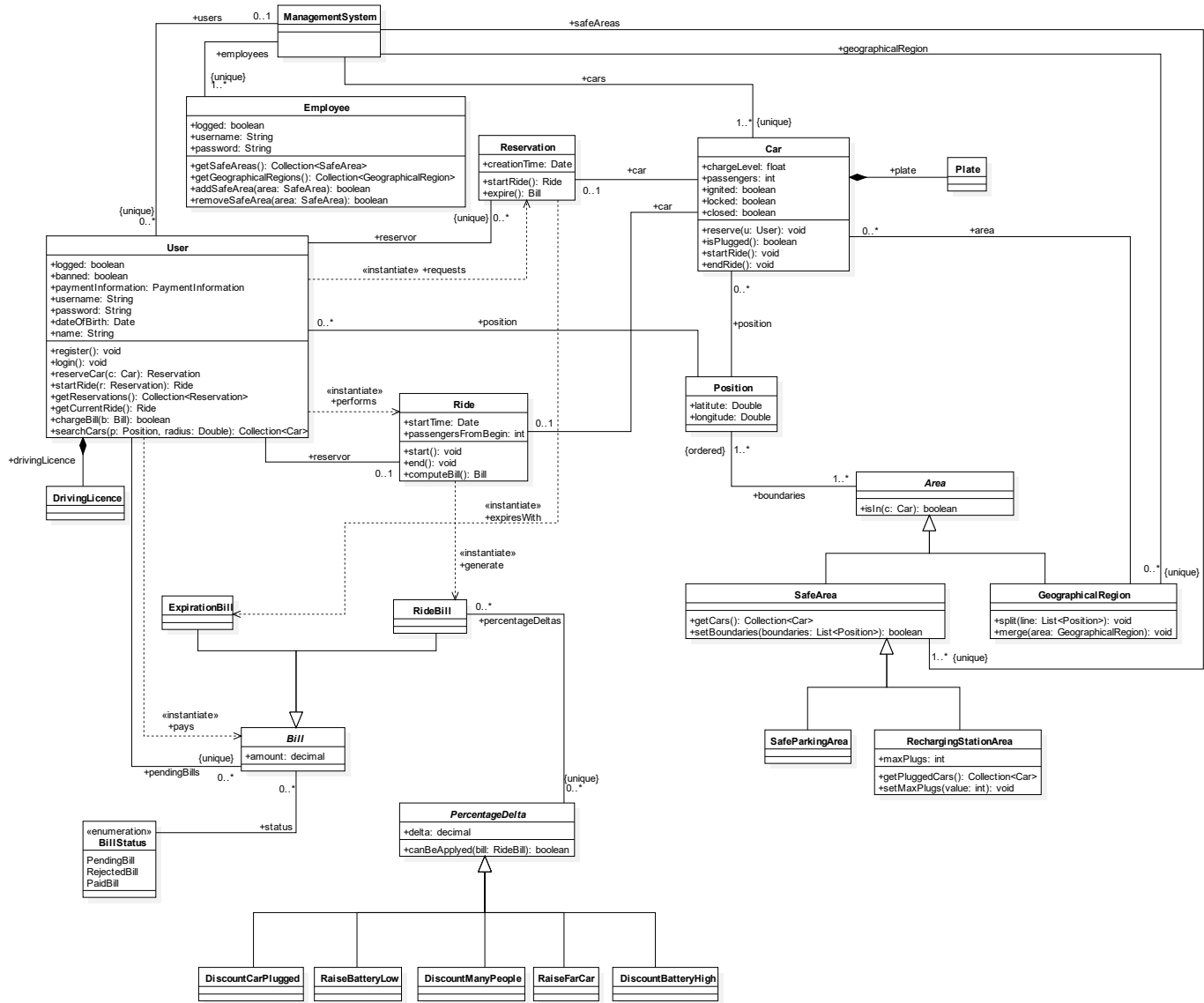
State charts
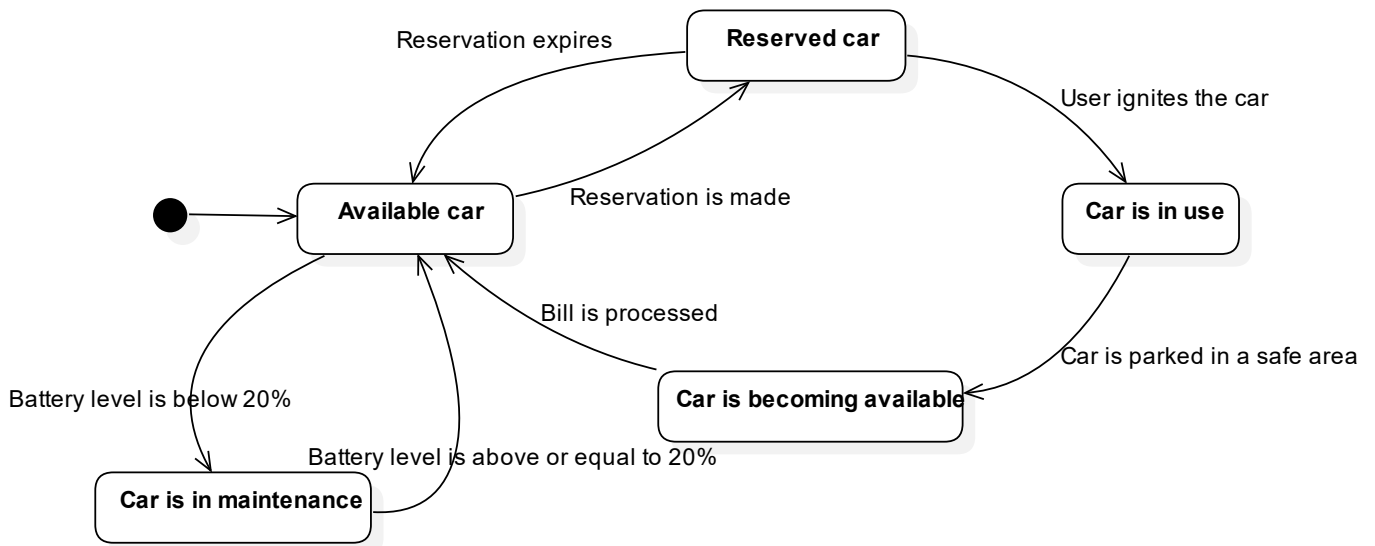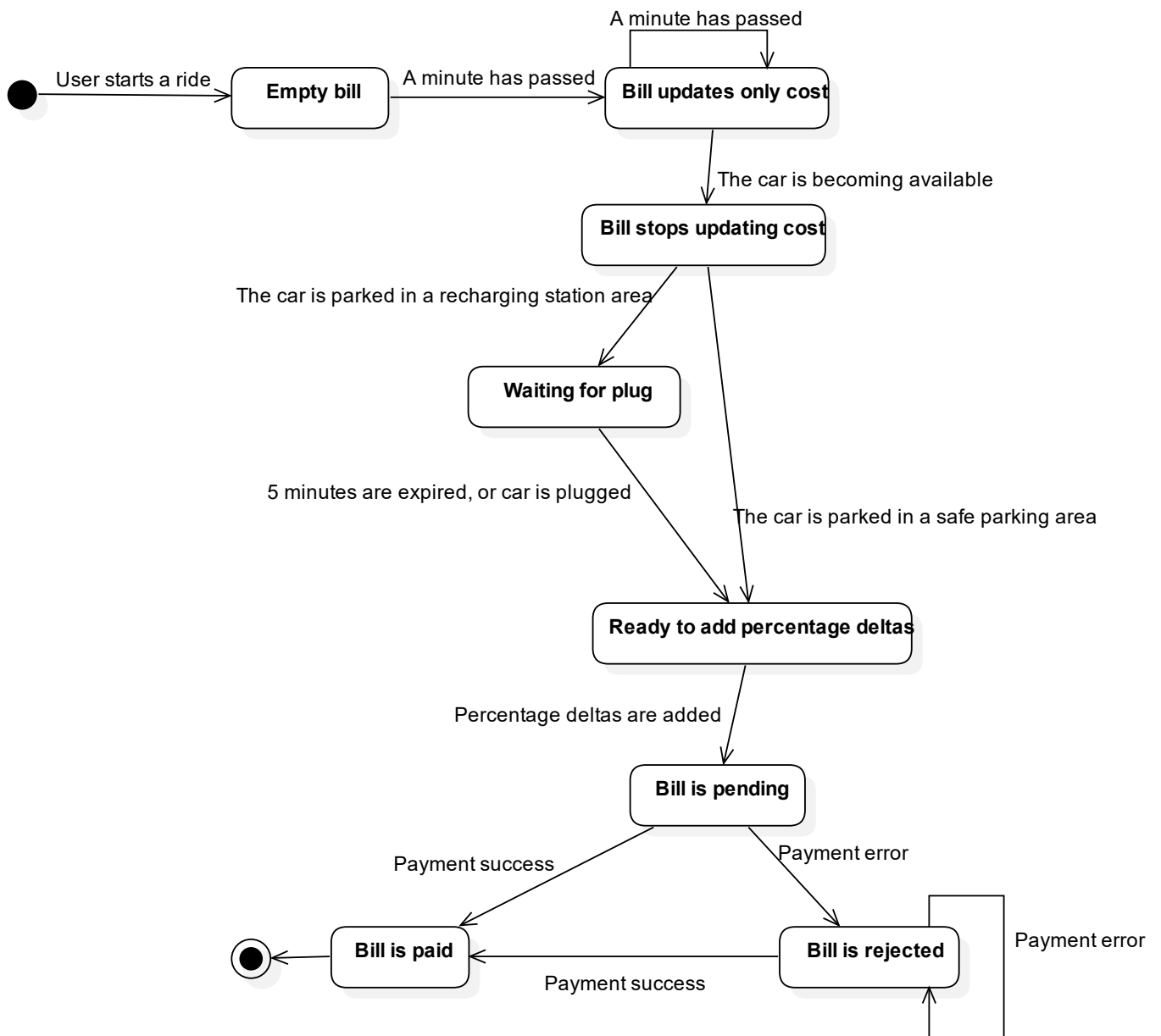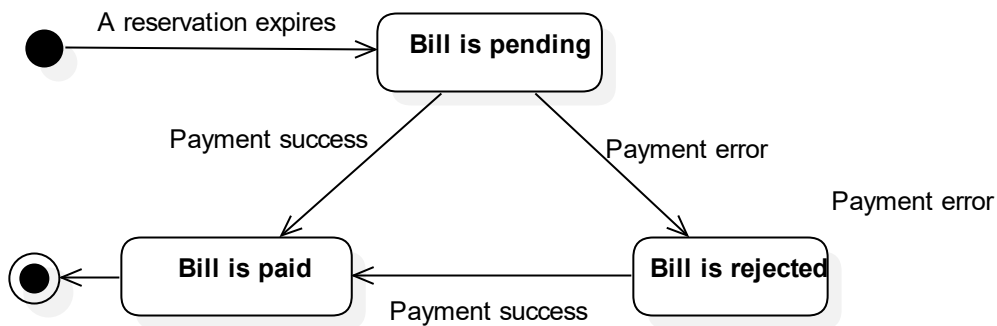
User state diagram
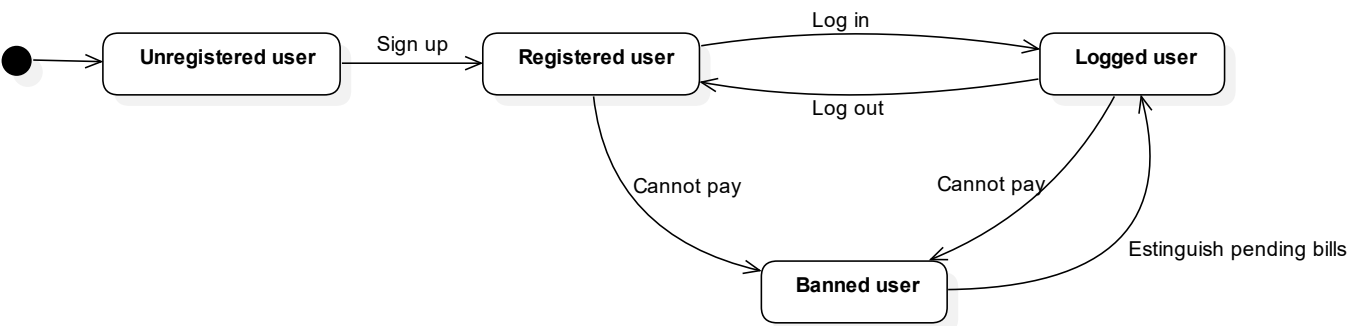
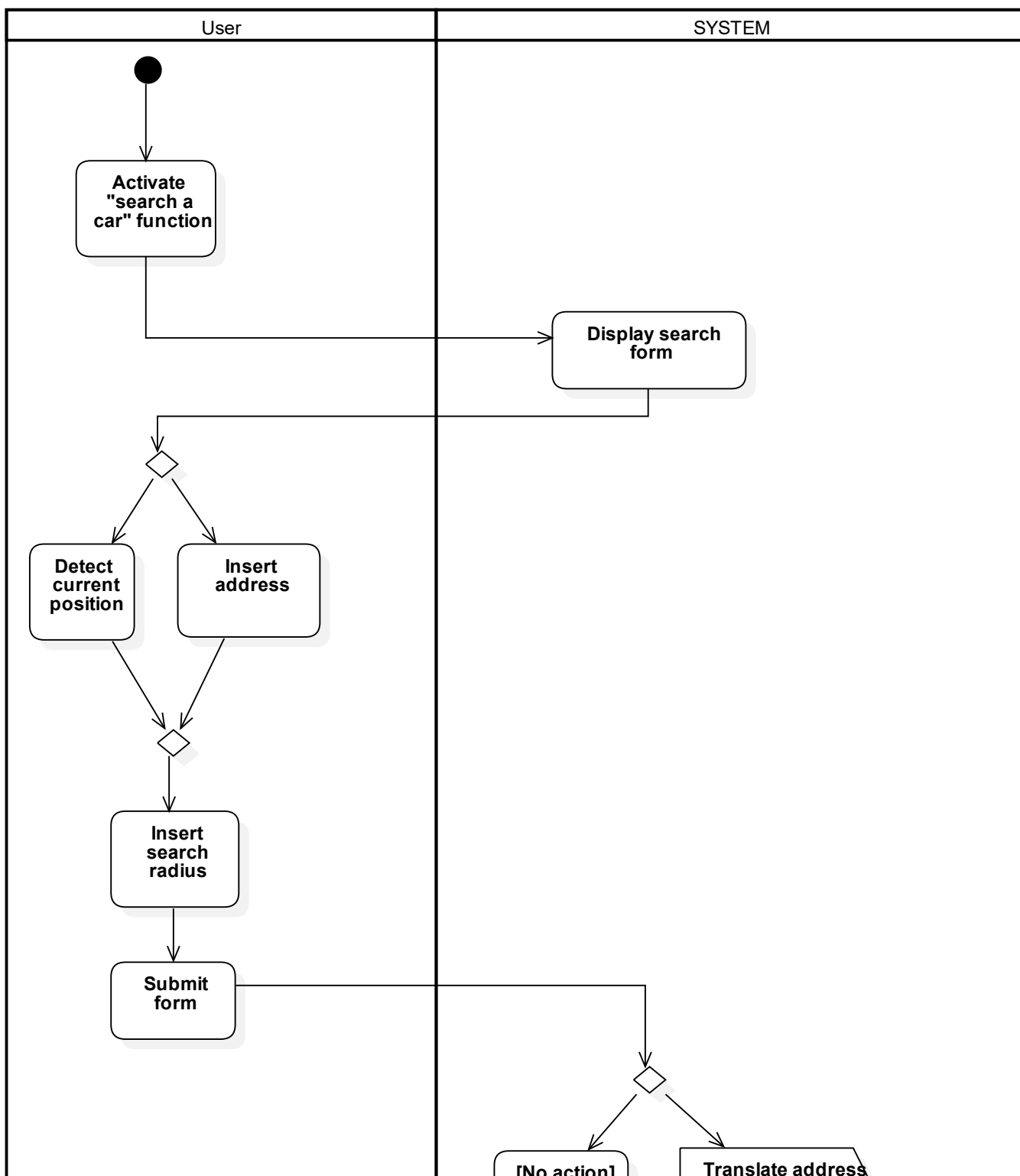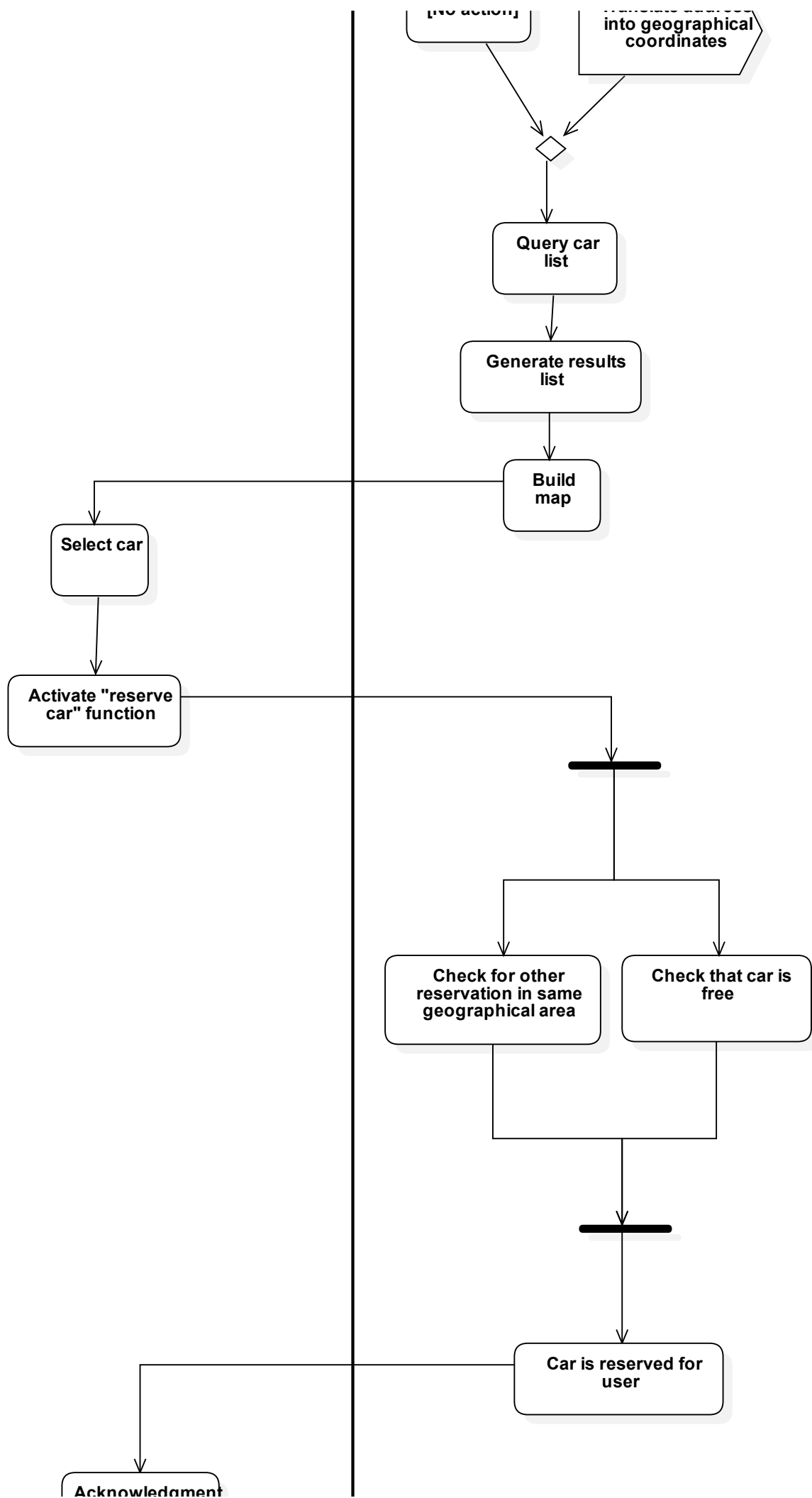### Car state model

Ride bill state diagram
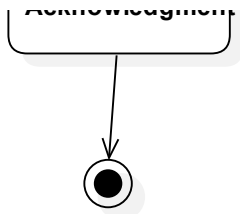

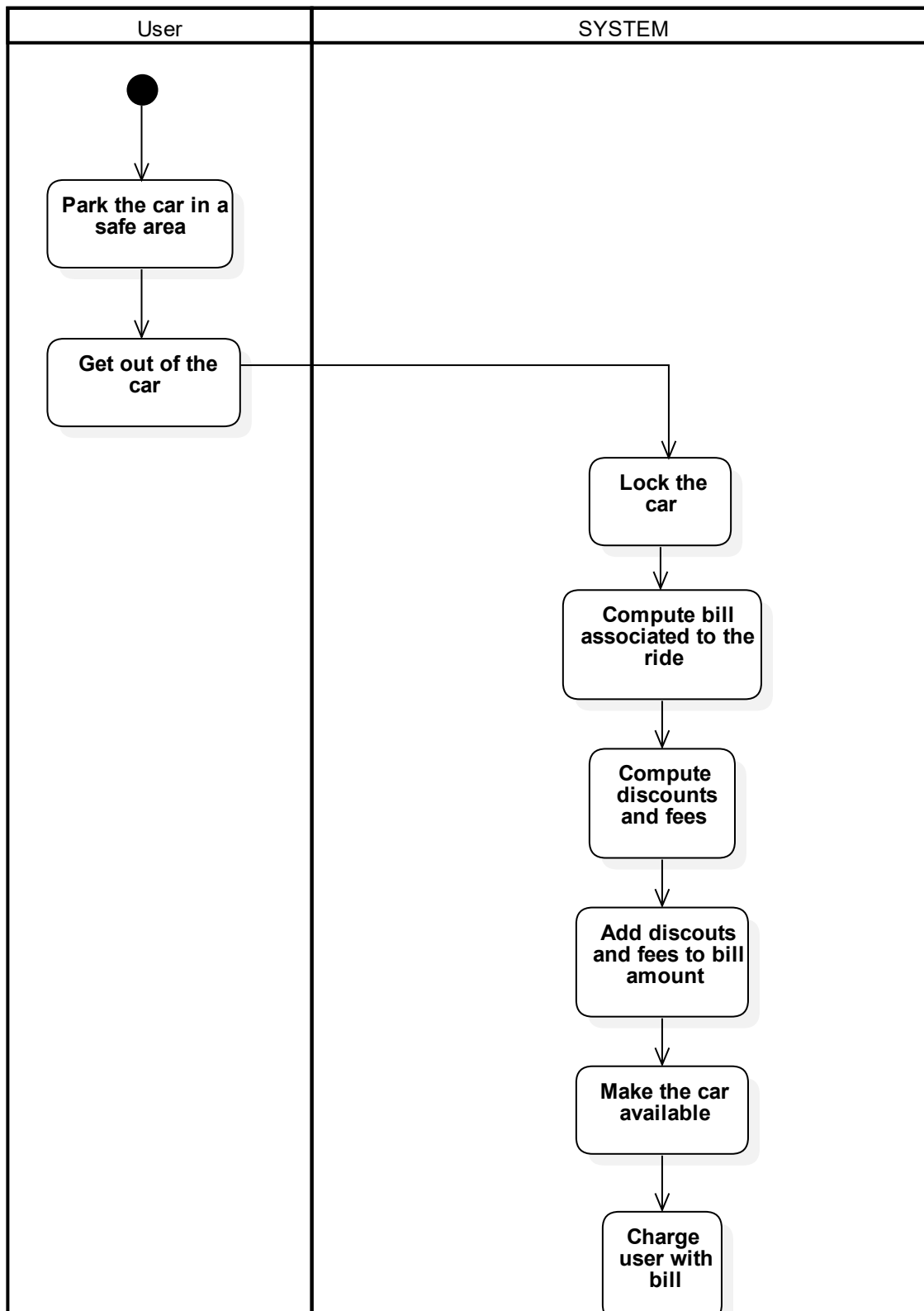## Expiration bill state diagram

Activity diagrams

Car search and reservation diagram
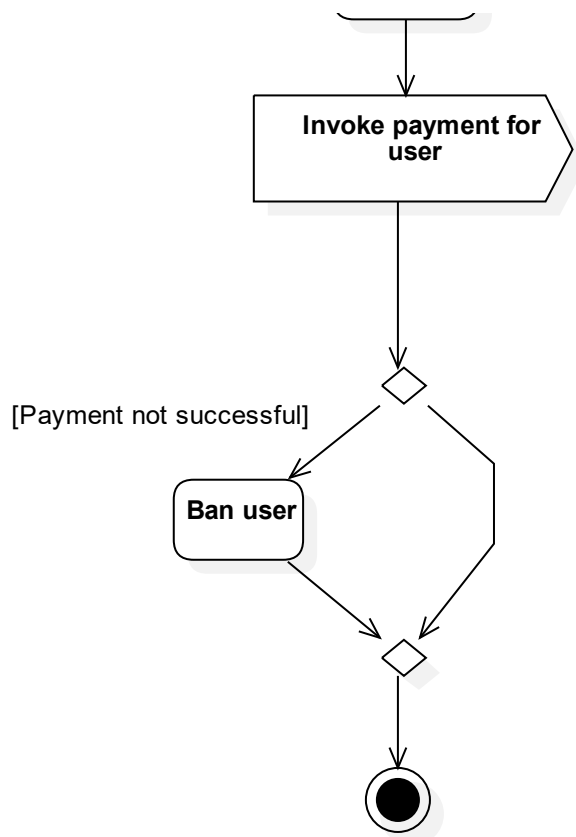
[No action]

Translate address into geographical coordinates

Query car list

Generate results list

Build map

Select car

Activate "reserve car" function

Check for other reservation in same geographical area

Check that car is free

Car is reserved for user

Acknowledgment

Acknowledgment

Ride end diagram

User	SYSTEM
Park the car in a safe area	
Get out of the car	Lock the car
	Compute bill associated to the ride
	Compute discounts and fees
	Add discouts and fees to bill amount
	Make the car available
	Charge user with bill

Sequence diagrams

Ride sequence diagram
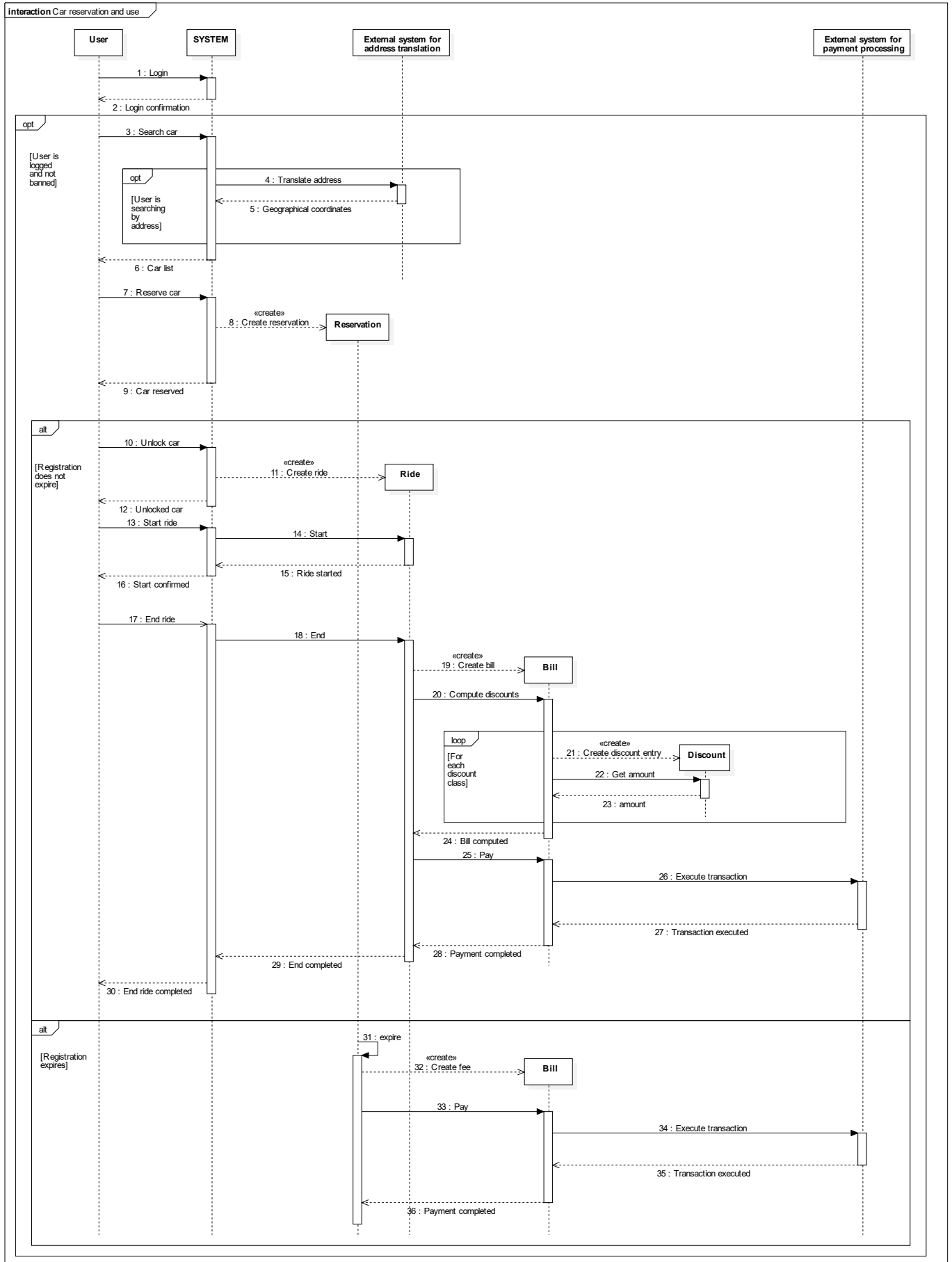
3.2.4 Application mockups

User related views

Sign-up view

Login view

Username

Password

LOGIN   or   SIGN UP

Search car near user's position view

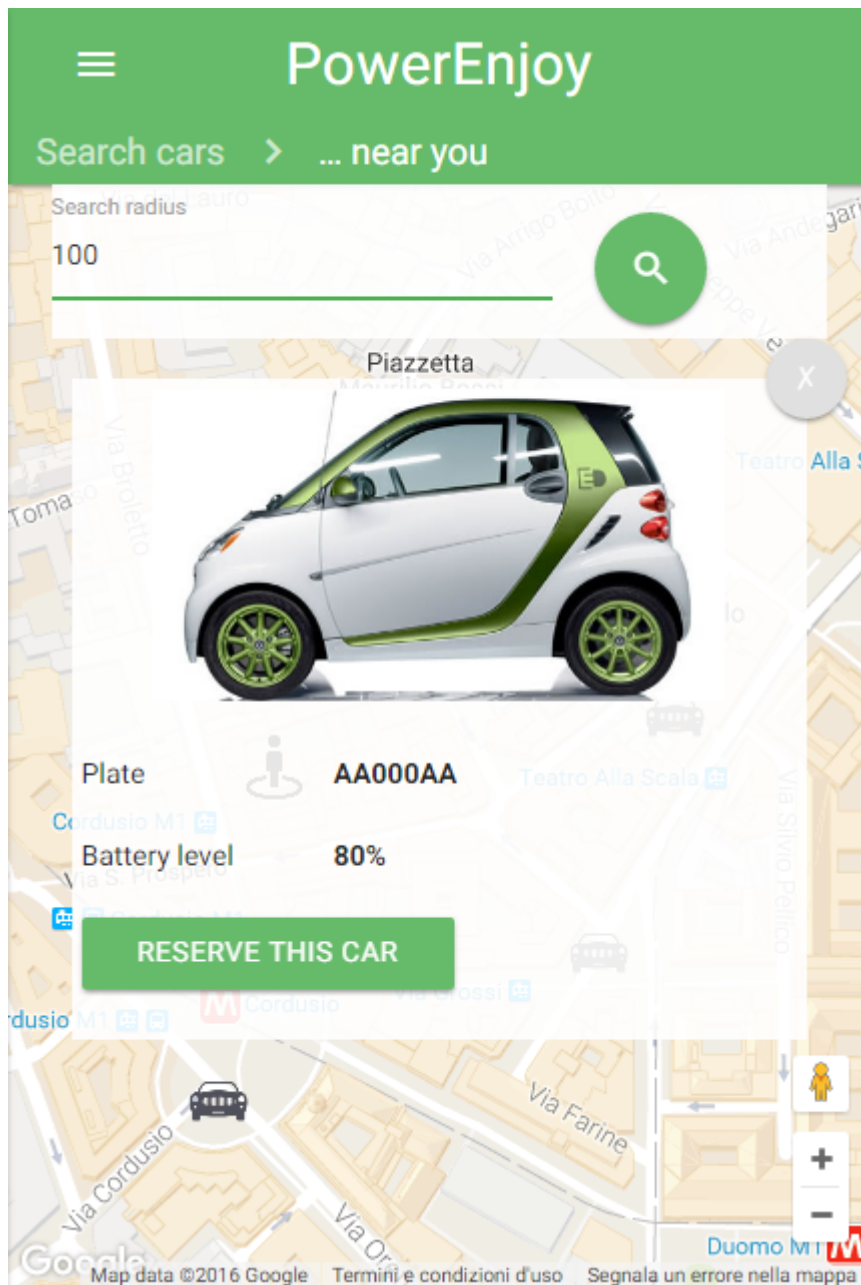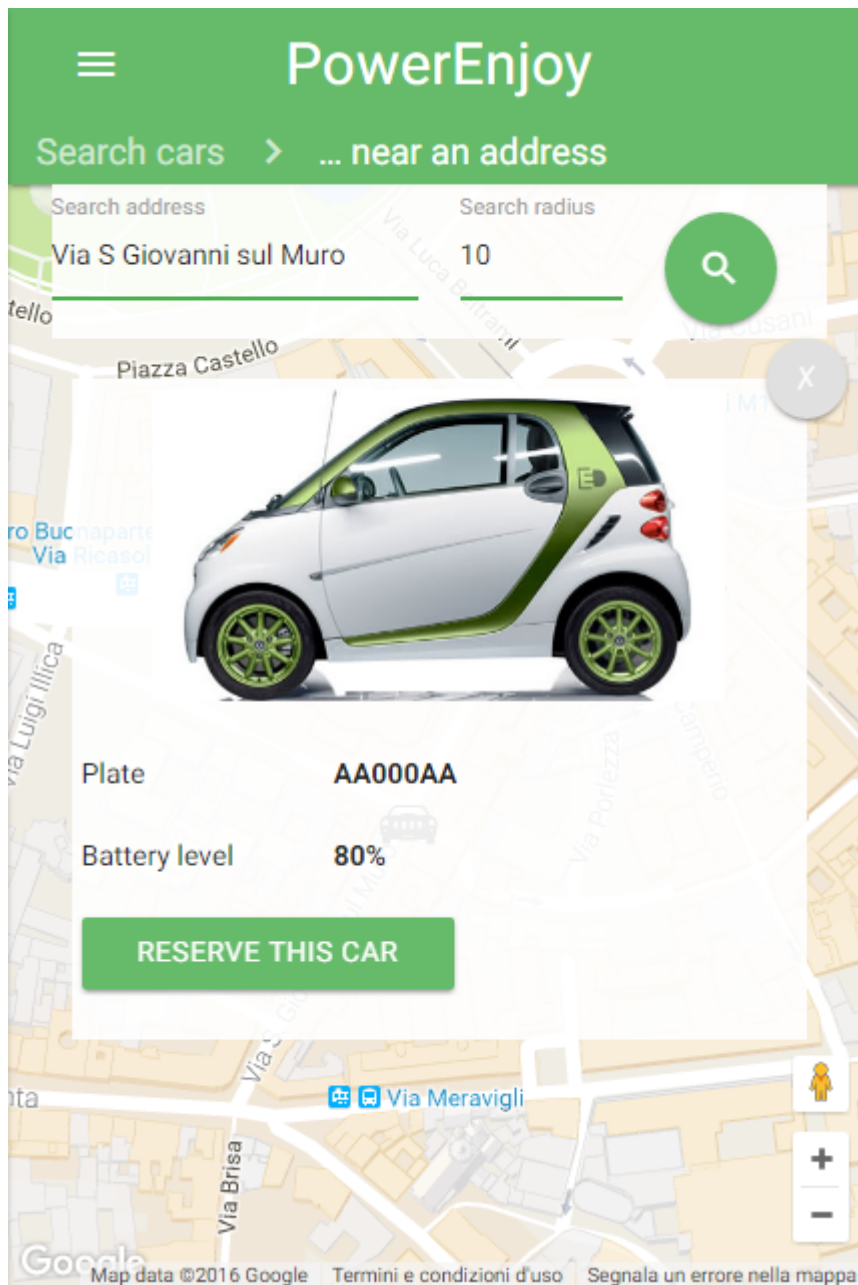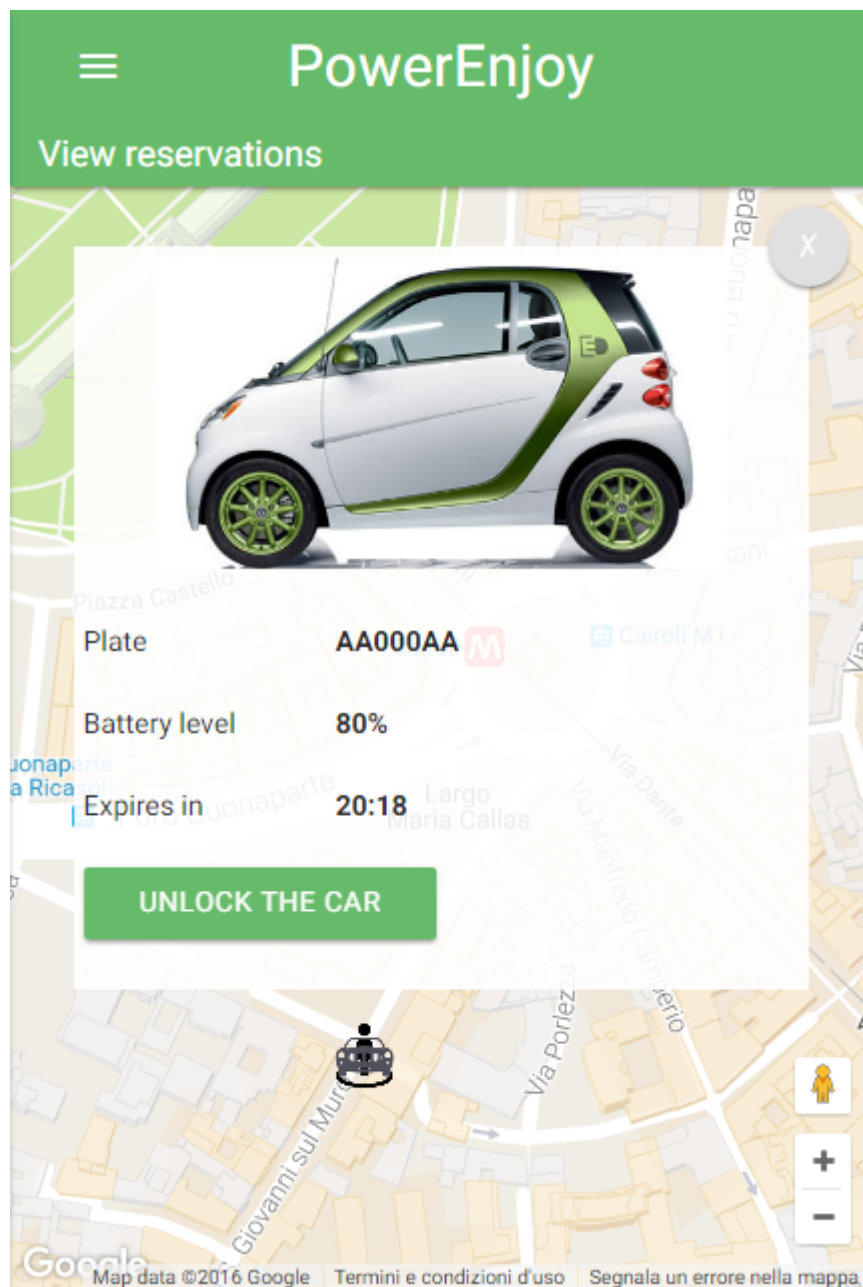**Search car near address view**

"My reservations" view

Pending payments view

Employee reserved view

Manage safe areas view
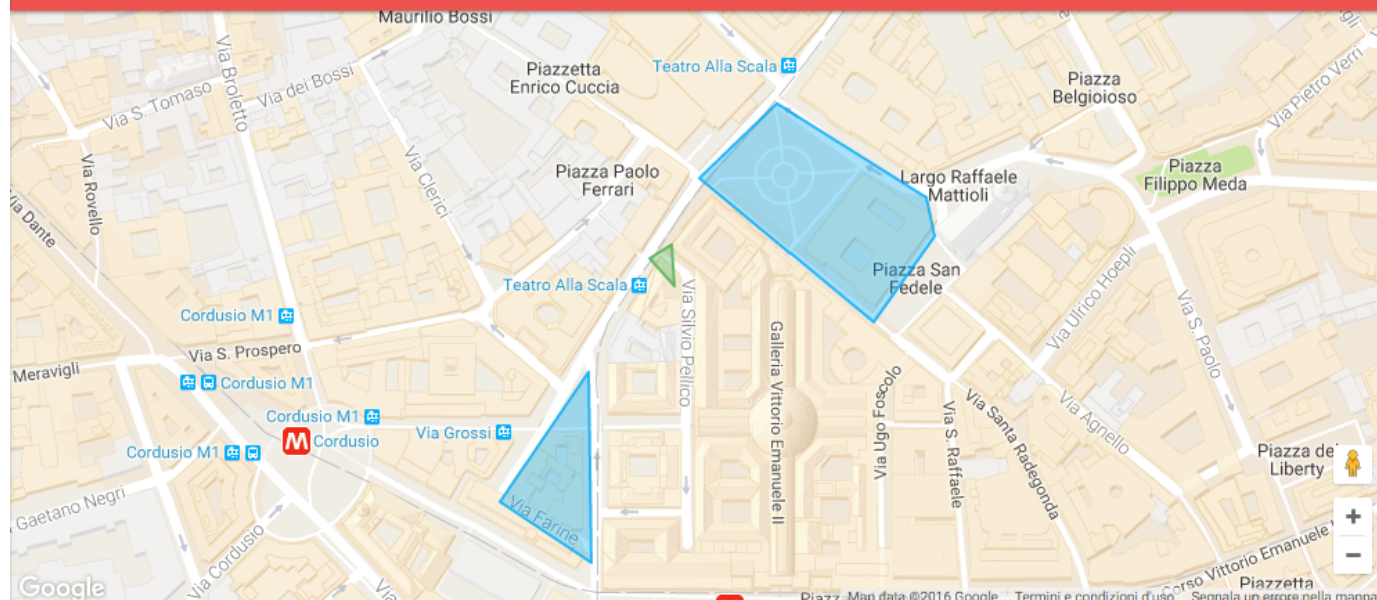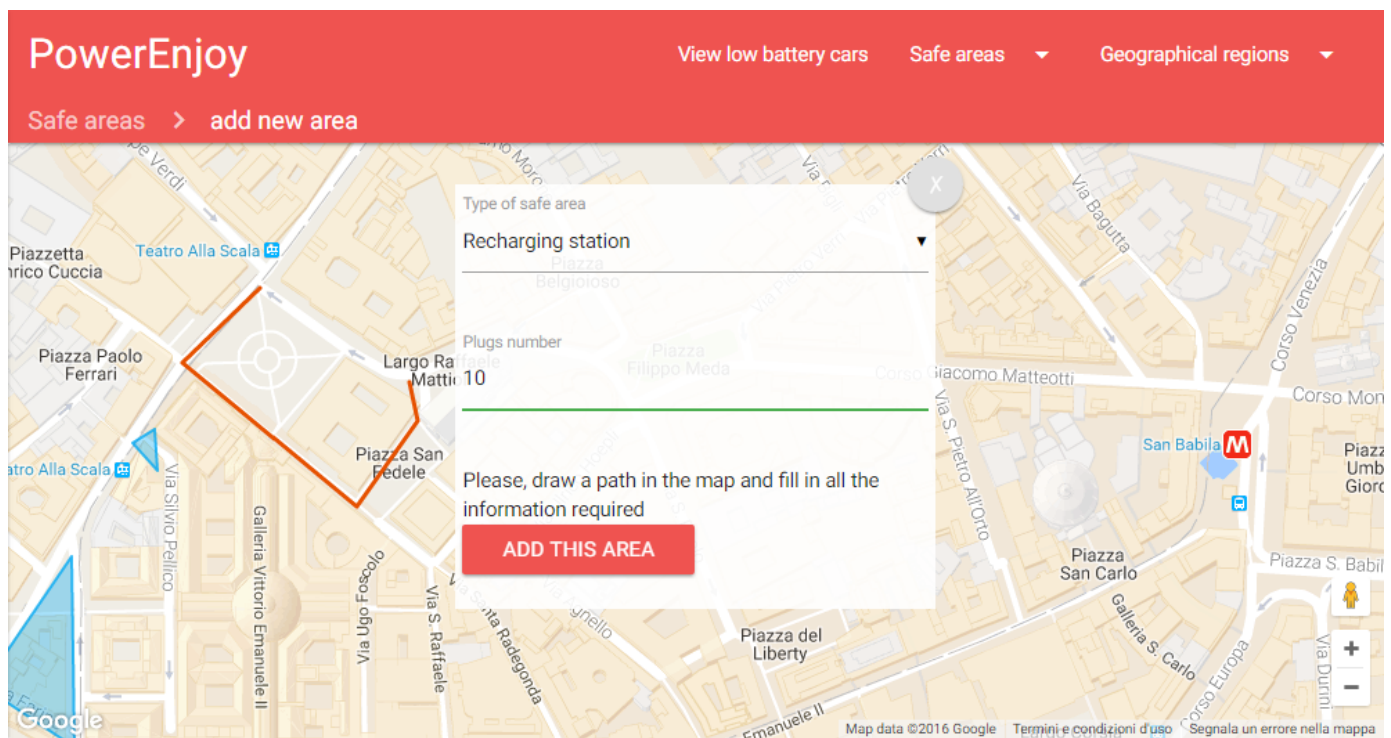
Add safe area view

Remove safe area view
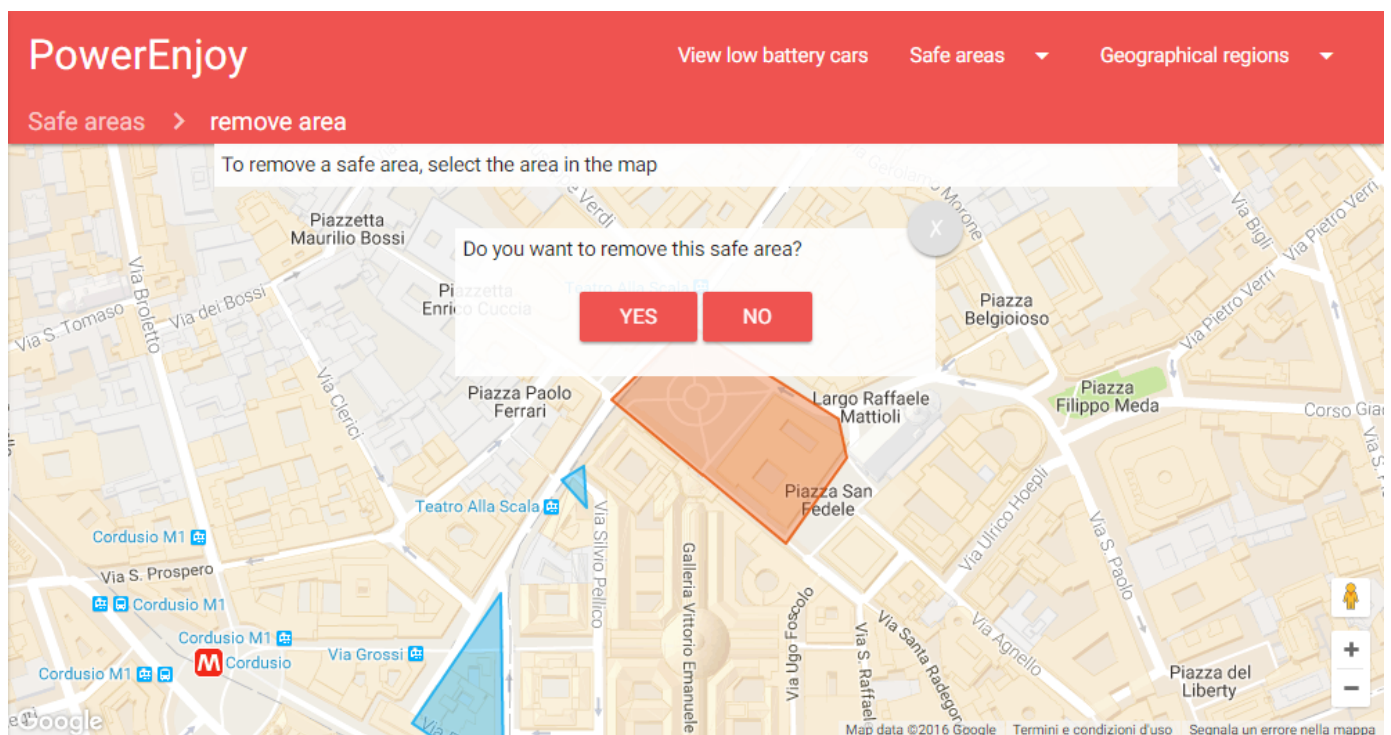
Update safe area view
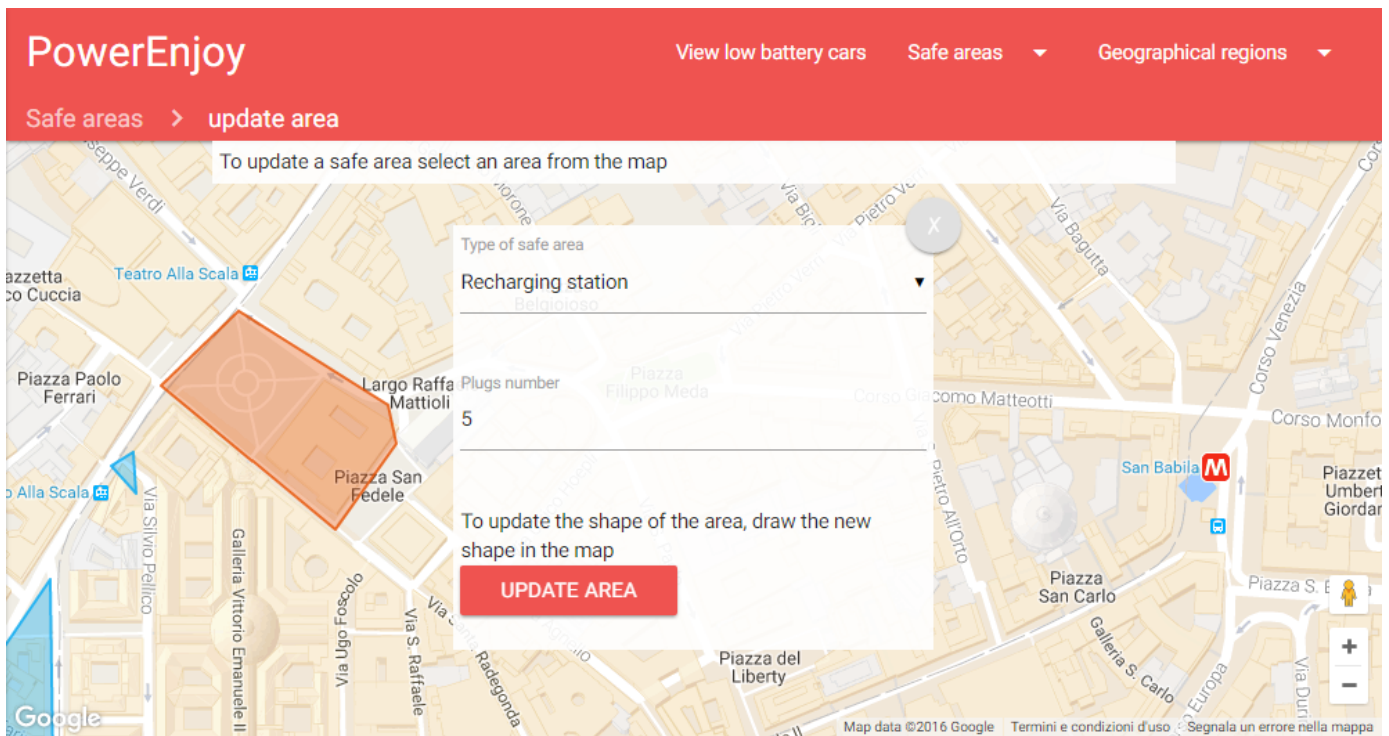
Manage geographical regions view
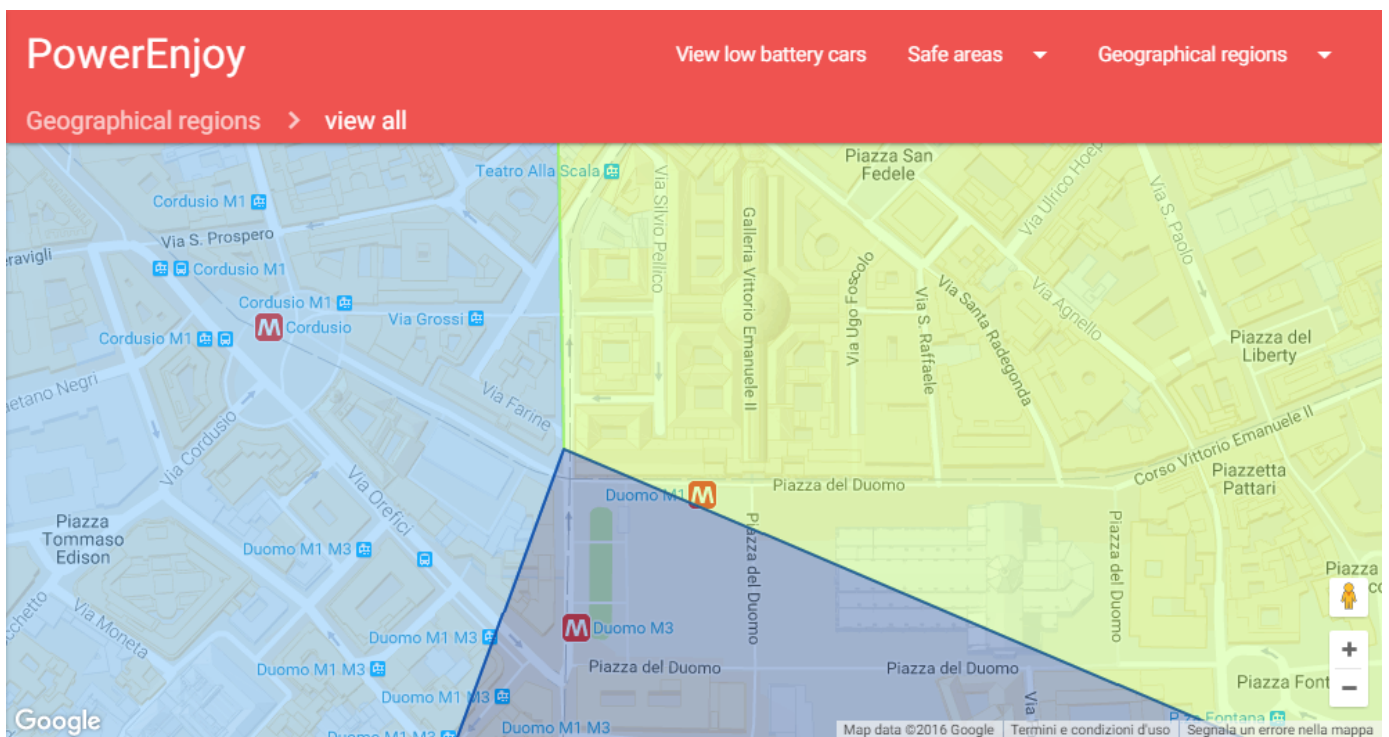
Split geographical region view
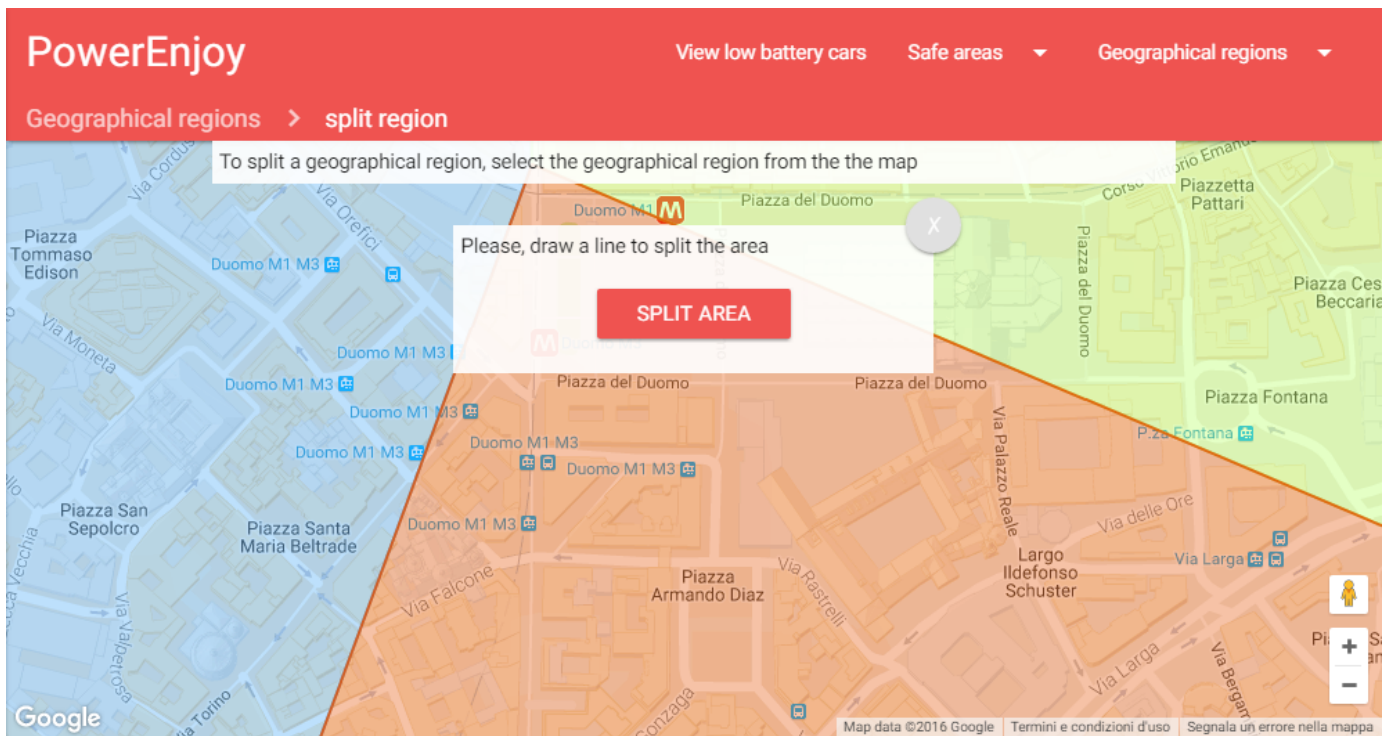
Merge geographical regions view
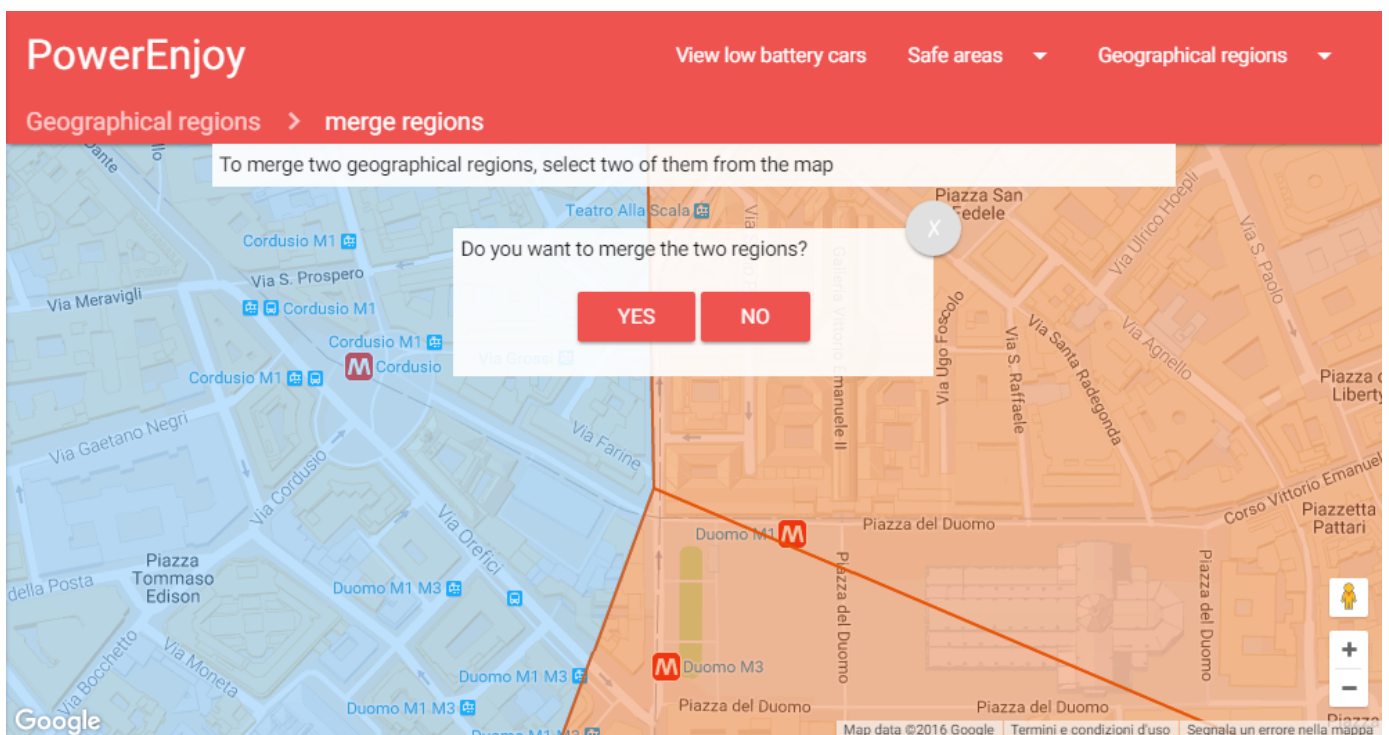
3.2.5 Specific requirements

- [G1] Users must be able to sign up to the system entering their name, birth date, address, credit card information and a valid driving license number, in addition to username and password:
    - [D1] Driving license number can be checked by an external service
    - [D14] A user interacting with the system has an internet connection
    - [R1.1]: The system can acquire user information for the registration (name, surname, address, birth date, driving licence number, credit card number and CVV)

- [R1.2]: The system validates the driving licence number using the external service mentioned in [D1]
- [R1.3]: The system validates the payment information using the external service mentioned in [D2]
- [R1.4]: The system is able to verify that no other registered user exists with the same username or driving licence number or payment information
- [R1.5]: The system registers this new user only if given information are valid

- [G2] Only registered users and employee must be able to log in to the system:
  - [D14] A user interacting with the system has an internet connection
  - [D17] Each employee has its own username and password (different from those used to reserve cars)
  - [R2.1]: The system can acquire user information for login (username and password)
  - [R2.2]: System is able to check whether a tuple (username, password) is correct, that is whether that tuple matches the information of a registered user or of an employee or not
  - [R2.3]: The user or the employee logs in if and only if username and password costitute a correct tuple

- [G3] Logged users must be able to find the location and the level of the battery of available cars within a user defined distance from his current location:
  - [D5] Car can communicate with the system via mobile internet connection
  - [D6] Car position is detected using GPS
  - [D7] The level of charge of the battery is detected by the system
  - [D13] The user position is detected using GPS
  - [D14] A user interacting with the system has an internet connection
  - [R3.1]: The system only allows the logged in user who is not banned to insert the search radius
  - [R3.2]: The system is capable of finding all available cars within the inserted distance from the user's position
  - [R3.3]: The system is able to show to a user a list of cars with their position and battery level

- [G4] Logged users must be able to find the location and the level of the battery of available cars within a user defined distance from a user defined address:
  - [D3] An external service is available to get the geographical coordinates of an address
  - [D5] Car can communicate with the system via mobile internet connection
  - [D6] Car position is detected using GPS
  - [D7] The level of charge of the battery is detected by the system
  - [D14] A user interacting with the system has an internet connection
  - [R3.1]
  - [R4.1]: The system only allows the logged in user who is not banned to insert the address on which the search area will be centered

- - [R4.2]: The system is capable of finding all available cars within a distance range from the geographical coordinate of the address
    - [R3.3]
- [G5] Logged users must be able to reserve for each geographical region at most one car among the available ones:
    - [D5] Car can communicate with the system via mobile internet connection
    - [D6] Car position is detected using GPS
    - [D7] The level of charge of the battery is detected by the system
    - [D14] A user interacting with the system has an internet connection
    - [D15] Geographical regions do not overlap
    - [R5.1]: The system only allows the logged in user who is not banned to reserve an available car
    - [R5.2]: The system is able to get the geographical region from the car geographical coordinates
    - [R5.3]: The system only reserves a car if the logged in user that requests it has no other reservation for the same geographical area in which the car is located
- [G6] The reservation of a car expires after one hour:
    - [R6.1]: The system keeps track of the time elapsed since a reservation is made
    - [R6.2]: If the elapsed time is greater than one hour, then the reservation expires
- [G7] Only the reservor user must be able to unlock the car he has a reservation for:
    - [D5] Car can communicate with the system via mobile internet connection
    - [D6] Car position is detected using GPS
    - [D11] If the system unlocks a car, anyone can enter the car
    - [D13] The user position is detected using GPS
    - [D14] A user interacting with the system has an internet connection
    - [R7.1]: The system unlocks a car only if distance between the car and the reservor user is less then 8 meters and the reservor user has requested the unlocking
- [G8] When a user ignites the engine of a car, the system starts computing the bill:
    - [D5] Car can communicate with the system via mobile internet connection
    - [D10] The system knows whether the engine is ignited or not
    - [R8.1]: The system can create an empty bill for the reservor user only when the engine is ignited
- [G9] As long as a car remains in use, the amount of the bill of the reservor user of that car is increased for a per-car defined amount of money per minute:
    - [D5] Car can communicate with the system via mobile internet connection
    - [D6] Car position is detected using GPS
    - [D8] The number of people in the car is detected by the system

- [D10] The system knows whether the engine is ignited or not
- [R9.1]: The system knows whether a car is in a safe area or not
- [R9.2]: The system knows the time elapsed since the engine ignition
- [R9.3]: The system updates the reservor user bill according to the elapsed time
- [G10] Car drivers are notified of the current bill through a screen on the car:
  - [D4] Each car has a display capable of showing the current bill amount
  - [R10.1]: The system knows the reservor user bill
- [G11] When a car is becoming available, then the bill can be charged to the reservor user:
  - [D2] There exists an external service for processing payments
  - [D5] Car can communicate with the system via mobile internet connection
  - [D6] Car position is detected using GPS
  - [D8] The number of people in the car is detected by the system
  - [D10] The system knows whether the engine is ignited or not
  - [D16] Safe areas do not overlap
  - [D18] The system is able to detect whether the doors of a car are opened or closed
  - [R9.1]
  - [R12.1]: When a car is becoming available and this car is parked in a recharging station area there is a 5 minute window before the system charges the bill to the reservor user
  - [R12.2]: When a car is becoming available and this car is parked in a safe parking area the system charges immediatly the reservor user with the bill
- [G12] The system must incentivize the virtuous behaviors of the users:
  - [D5] Car can communicate with the system via mobile internet connection
  - [D6] Car position is detected using GPS
  - [D7] The level of charge of the battery is detected by the system
  - [D8] The number of people in the car is detected by the system
  - [D9] The system knows whether a car has been plugged for recharge or not
  - [D10] The system knows whether the engine is ignited or not
  - [R16]
  - [D18] The system is able to detect whether the doors of a car are opened or closed
  - [R11.1]: When a car is becoming available, if the system detects that this car has had at least two passengers (not including the driver) for all the time of the ride when the engine was ignited, then the system applies a discount of 10% on the bill of the reservor user
  - [R11.2]: When a car is becoming available, if its battery level is greater or equal to the 50% of the total battery level, the system applies a discount of 20% on the bill of the reservor user
  - [R11.3]: When a car is becoming available, if it is parked in a recharging station area and it is plugged to the power grid, the system applies a discount of 30% on the bill of the reservor user and the 5 minute window terminates

- [R11.4]: When a car is becoming available, if the battery level is less than 20% of the total battery level, the system applies a raise of 30% on the bill of the reservor user
    - [R11.5]: When a car is becoming available, if it is left more than 3Km away from the nearest recharging station, the system applies a raise of 30% on the bill of the reservor user
- [G13] After being used by its reservor user, a car will be available for a new reservation after the system has charged that user with the bill corresponding to his ride:
    - [D2] There exists an external service for processing payments
    - [D5] Car can communicate with the system via mobile internet connection
    - [D12] If the system locks a car, nobody can enter the car
    - [R13.1]: The system knows whether a payment attempt was made for the current car
    - [R13.2]: The system locks the car and makes it available only if the payment attempt was processed, either if the payment is successful or not
- [G14] A user that cannot pay his bill is banned until the bill is estinguished:
    - [D2] There exists an external service for processing payments
    - [R14.1]: The system knows the result of the payment operation
    - [R14.2]: The user is banned only if there exists a pending bill bound to him
    - [R14.3]: The system marks a bill as paid if and only if a payment operation associated to this bill is successful
    - [R14.4]: A user can ask the system to try again to extinguish his pending bills
    - [R14.5]: Only pending bills can be required to be paid
- [G15] Logged users must be able to find all their reservation that are still valid:
    - [D5] Car can communicate with the system via mobile internet connection
    - [D6] Car position is detected using GPS
    - [D7] The level of charge of the battery is detected by the system
    - [R15.1]: The system only allows a logged in user who is not banned to view his reservations
    - [R15.2]: The system is capable of finding all the reservation made by the user
    - [R15.3]: The system is able to show to a user a list of cars with their position, battery level and the expiration time
- [G16] Logged employees are able to view all the geographical regions:
    - [R16.1]: The system only allows a logged in employee to manage geographical regions
    - [R16.2]: The system is able to find all the geographical regions already defined
    - [R16.3]: The system is able to display the geographical regions to the employee
- [G17] Logged employees are able to split geographical regions:
    - [R16.1]
    - [R16.2]
    - [R16.3]

- [R17.1]: The system allows the employee to select a geographical region
- [R17.2]: The system allows the employee to draw a line inside the selected geographical region
- [R17.3]: The system is able to compute the new geographical region and store them
- [G18] Logged employees are able to merge geographical regions:
  - [R16.1]
  - [R16.2]
  - [R16.3]
  - [R18.1]: The system allows the employee to select two geographical regions
  - [R18.2]: The system merges the two region into one single region and store it, removing the two sources region
- [G19] Logged employees are able to view all the safe areas:
  - [R19.1]: The system only allow a logged in employee to manage safe areas
  - [R19.2]: The system is able to find all the safe areas already defined
  - [R19.3]: The system is able to display the safe areas to the employee
- [G20] Logged employees are able to remove a safe area:
  - [R19.1]
  - [R19.2]
  - [R19.3]
  - [R20.1]: The system allows the employee to select a safe area
  - [R20.2]: The system remove the selected safe area
- [G21] Logged employees are able to add a new safe area:
  - [R19.1]
  - [R19.2]
  - [R19.3]
  - [R21.1]: The system can acquire from the employee the type of safe area, its shape as a sequence of coordinate, and eventually the number of plugs of the recharging station area
  - [R21.2]: The system is able to check if the defined safe area will overlap with the already defined safe areas
  - [R21.3]: The system insert the new safe area only if it is not overlapping
- [G22] Logged employees are able to update the information of a safe area:
  - [R19.1]
  - [R19.2]
  - [R19.3]
  - [R22.1]: The system allows the employee to select a safe area
  - [R21.1]: The system can acquire from the employee the type of safe area, its shape as a sequence of coordinate, and eventually the number of plugs of the recharging station area

- [R22.2]: The system is able to check if the defined safe area will overlap with the already defined safe areas except for the selected one
- [R22.3]: The system updates the selected safe area only if it is not overlapping
- [G23] Logged employees can view all the cars which are in mantainace and are not plugged into a power grid:
  - [D5] Car can communicate with the system via mobile internet connection
  - [D6] Car position is detected using GPS
  - [D7] The level of charge of the battery is detected by the system
  - [R23.1]: The system only allows a logged in employee to view the list of in maintenance cars
  - [R23.2]: The system is capable of finding all the in maintenance cars
  - [R3.3]

3.3 Performance requirements

1. There is no limit to the number of registered users
2. All of the requests must be processed 10 seconds. The majority of them (>90%) must be processed in less than 5 seconds.
3. The system must be able to manage more than 100 simultaneos connections from users
4. The system must be able to handle all the connections with the cars in the same time. Signals coming from cars must be processed in less than 1 second, in order to grant a fluid interaction experience to the users when interacting with the cars.

3.4 Software system attributes

3.4.1 Reliability

System is designed to run on a single server. A backup server is also present, in order to take place of main server in case of failure. This is done to prevent service unavailability and to ensure that reserved cars can be effectively picked up by users.

3.4.2 Availability

The system must guarantee an availability of 99%.

3.4.3 Security

- All the communications between server and clients must be protected by using TLS protocol.

- All attempts of establishing an unsecure communication with the server must be refused.

- Users' passwords must be stored using a one way function

- Users' payment informations and driving licence number must be stored in a private location

3.4.4 Maintainability

All the code must be documented using JavaDoc so that other developers can easily understand and edit it. The system must provide a logging functionality for debug purpose.

3.4.5 Portability

The web application being written in Java, will be portable to any OS able to run the JVM and Java class files with major version 51. The interface with the employees and the users must be compatible with the last two years version of the major browsers that are Android Browser, Chrome, Chrome for Android, Edge, Firefox, Firefox for Android, IE Mobile, Internet Explorer, Safari, iOS Safari.

3.5 Alloy

World model

```
1   /*
2       Simplification,an area can be any kind of polygon, but for what we are
3       interested in, an area is a set of point inside such polygon
4   */
5   abstract sig Area {
6       boundaries: some Position
7   } {
8       #boundaries>2
9   }
10  sig GeographicalRegion extends Area {}
11  abstract sig SafeArea extends Area {}
12  sig SafeParkingArea extends SafeArea {}
13  sig RechargingStationArea extends SafeArea {
14      maxPlugs: one Int,
15      pluggedCars: set Car
16  } {
17      maxPlugs > 0
18      #pluggedCars <= maxPlugs
19      all c:pluggedCars| c.position in boundaries
20  }
21  /*
22      It finds a position with a minimal distance from the argument, such that
23      the position found belongs to a RechargingStationArea
24  */
25  fun findNearestPositionOfRechargingStations[pos:Position]:lone Position {
26      {p:RechargingStationArea.boundaries|
27          all p2:RechargingStationArea.boundaries| p.distance[pos] <= p2.distance[pos]
28      }
29  }
30  /*
31      Geographical regions cannot overlap, because of the way a geographical region can
    be modified
32  */
```

```
33    fact geographicalRegionDoNotOverlap {
34        no disjoint g1,g2:GeographicalRegion| g1.boundaries&g2.boundaries != none
35    }
36    /*
37        Safe areas cannot overlap because the two subset are defined to be partitions of
      the superset.
38        Overlapping within the same partition, even if logically possible, it is useless.
39    */
40    fact safeAreaDoNotOverlap {
41        no disjoint s1,s2:SafeArea| s1.boundaries&s2.boundaries != none
42    }
43    /*
44        Since geographical region must cover all the territory, it is necessary that any
      position belongs
45        to a geographical area
46    */
47    fact eachPositionBelongsToAtLeastGeographicalRegion {
48        all p:Position| p in GeographicalRegion.boundaries
49    }
50    assert reservationOnRideCar {
51        no r:Reservation,rr:Ride| r.car = rr.car
52    }
53    assert noDoubleRide {
54        no disjoint r1,r2: Ride| r1.reservor = r2.reservor
55    }
56    assert positionBelongsToOneRegion {
57        no disjoint r1,r2: GeographicalRegion| (r1.boundaries & r2.boundaries) != none
58    }
59    assert noTwoPendingRideBill {
60        no u:User| #({p:u.pendingBills| p in RideBill and p.status = PendingBill})>1
61    }
62    assert onRejectedNoReservation {
63        no r:Reservation| {b:r.reservor.pendingBills| b.status = RejectedBill} != none
64    }
65    assert noOverFreeBonusCombo {
66        no r:Ride| {
67            sumPercentageDelta[{p:PercentageDelta| canApplyPercentageDelta[r,p]}]<-100
68        }
69    }
70    assert noDoubleReservation {
71        no disjoint r1,r2: Ride| r1.car = r2.car
72    }
73    check reservationOnRideCar for 5 but 8 Int
74    check noDoubleRide for 5 but 8 Int
75    check positionBelongsToOneRegion for 5 but 8 Int
76    check noTwoPendingRideBill for 5 but 8 Int
77    check onRejectedNoReservation for 5 but 8 Int
78    check noOverFreeBonusCombo for 5 but 8 Int
79    check noDoubleReservation for 5 but 8 Int
80    enum BillStatus { PendingBill, PaidBill, RejectedBill }
81    abstract sig Bill {
82        amount: one Int,
83        status: one BillStatus
84    }
85    sig ExpirationBill extends Bill {} {
86        amount = 1
87    }
88    sig RideBill extends Bill {
89        percentageDeltas: set PercentageDelta,
90    } {
91        amount >= 0
92        status = PendingBill => {
93            one r:Ride| {
94                this in r.reservor.pendingBills
95                r.car.isBecomingAvailable[] => {
96                    all p:PercentageDelta| canApplyPercentageDelta[r, p] <=> p in
      percentageDeltas
97                    amount = div[mul[mul[r.elapsedMinutes,r.car.costPerMinute], 100+
    sumPercentageDelta[percentageDeltas]],100]
```

```
 98                 }
 99                 not r.car.isBecomingAvailable[] => {
100                     percentageDeltas = none
101                     amount = mul[r.elapsedMinutes, r.car.costPerMinute]
102                 }
103             }
104         }
105         status = RejectedBill => amount != 0
106 }
107 fact thereCannotTwoRideBillsOneIsPending {
108     no rb1,rb2:RideBill| {
109         rb1.status = PendingBill
110         rb2.status = RejectedBill
111     }
112 }
113 fact allBillAreUsed {
114     User.pendingBills = Bill
115 }
116 fact billAreUniqueByUser {
117     no disjoint u1,u2:User| {
118         u1.pendingBills & u2.pendingBills != none
119     }
120 }
121 fact allRideHaveAnAssociatedBill {
122     all r:Ride| one b:RideBill| {
123         b.status = PendingBill
124         b in r.reservor.pendingBills
125     }
126 }
127 fact notStoringCompletedBills {
128     no b:Bill|b.status = PaidBill
129 }
130 abstract sig Boolean {}
131 one sig True extends Boolean {}
132 one sig False extends Boolean {}
133 sig Plate {}
134 sig Car {
135     plate: one Plate,
136     ignited: one Boolean,
137     passengers: one Int,
138     locked: one Boolean,
139     closed: one Boolean,
140     position: one Position,
141     costPerMinute: one Int,
142     chargeLevel: one Int,               //percentage
143     area: one GeographicalRegion
144 } {
145     chargeLevel >= 0
146     chargeLevel <= 100
147     passengers >= 0
//passengers are persons... and include the driver
148     costPerMinute > 0                   //negative or zero cost ride are not allowed
149     position in area.boundaries         //the position must belong to the area
150     locked = True => {
151         passengers = 0
152         ignited = False
153         closed = True
154         position in SafeArea.boundaries
155     }
156     chargeLevel = 0 => ignited = False
157 }
158 pred Car.isBecomingAvailable[]{
159     this.ignited=False
160     this.closed=True
161     this.locked=False
162     this.passengers = 0
163     this.position in SafeArea.boundaries
164 }
165 fact platesAreUniqueByCar {
```

```alloy
166        no disjoint c1,c2:Car| c1.plate = c2.plate
167    }
168    fact platesAreAllUsed {
169        Car.plate = Plate
170    }
171    fact onlyLockedCarCanBeRecharged {
172        no c:Car,rc:RechargingStationArea| c.locked = False and c in rc.pluggedCars
173    }
174    abstract sig PercentageDelta {
175        delta: one Int
176    } {
177        delta >= -100
178        delta <= 100
179    }
180    one sig DiscountManyPeople extends PercentageDelta {} {
181        delta = -10
182    }
183    one sig DiscountBatteryHigh extends PercentageDelta {} {
184        delta = -20
185    }
186    one sig DiscountCarPlugged extends PercentageDelta {} {
187        delta = -30
188    }
189    one sig RaiseFarCar extends PercentageDelta {} {
190        delta = 30
191    }
192    one sig RaiseBatteryLow extends PercentageDelta {} {
193        delta = 30
194    }
195    pred canApplyPercentageDelta[r: one Ride, p: one PercentageDelta] {
196        p = RaiseBatteryLow => r.car.chargeLevel < 20
197        p = RaiseFarCar => findNearestPositionOfRechargingStations[r.car.position].
    distance[r.position] > 9000000
198        p = DiscountCarPlugged => r.car in RechargingStationArea.pluggedCars
199        p = DiscountBatteryHigh => r.car.chargeLevel >= 50
200        p = DiscountManyPeople => r.passengersFromBegin >= 3
201    }
202    //consistency among deltas
203    fact BatteryIsLowOrHigh {
204        no b:Bill| {
205 &nbnbsp;        RaiseBatteryLow in b.percentageDeltas
206            DiscountBatteryHigh in b.percentageDeltas
207        }
208    }
209    fun sumPercentageDelta[s: set PercentageDelta]: one Int {
210        (DiscountManyPeople in s => DiscountManyPeople.delta  else 0)+
211        (DiscountBatteryHigh in s => DiscountBatteryHigh.delta else 0)+
212        (DiscountCarPlugged in s => DiscountCarPlugged.delta else 0)+
213        (RaiseFarCar in s => RaiseFarCar.delta else 0) +
214        (RaiseBatteryLow in s => RaiseBatteryLow.delta else 0)
215    }
216    one sig ManagementSystem {
217        users: set User,
218        cars: set Car,
219        safeAreas: set SafeArea,
220        geographicalRegion: set GeographicalRegion
221    } {
222        //users must be served
223        users!=none =>    {
224            cars!=none
225            (safeAreas & RechargingStationArea) != none
226        }
227        //areas must be complete
228        safeAreas = SafeArea
229        cars = Car
230        geographicalRegion = GeographicalRegion
231    }
232    /*
233        Represents a simplyfied version of a geographical position.
```

```
234      Its components are integer.
235  */
236  sig Position {
237      latitude: one Int,
238      longitude: one Int
239  }
240  /*
241      It computes the approximate square distance between two position.
242      It does not take into account the fact that the Earth is not a plane.
243  */
244  fun Position.distance[p: Position]: one Int {
245      mul[p.latitude-this.latitude, p.latitude-this.latitude]+mul[
     p.longitude-this.longitude,p.longitude-this.longitude]
246  }
247  /*
248      It has no sense to have two position with the same longitude and latitude.
249      Moreover, having such duplicates make comparsion of position more complicated
250  */
251  fact noDuplicatesInPosition{
252      no disjoint p1,p2:Position| {
253          p1.latitude=p2.latitude
254          p1.longitude=p2.longitude
255      }
256  }
257  /*
258      The elapsedMinutes relation is a simplification of the creationTime field in the
     class diagram.
259      The elapsedMinutes is the only relevant information for the system as it is, and
     can be computed
260      quite easly from the creationTime.
261  */
262  sig Reservation {
263      reservor: one User,
264      car: one Car,
265      elapsedMinutes: one Int
266  } {
267      elapsedMinutes >= 0
268      car.locked = True
269      car.chargeLevel > 20
270  }
271
272  fact reservationsMustExpire {
273      all r:Reservation| r.elapsedMinutes < 60
274  }
275  fact onlyRegisteredNotBannedUserCanHaveReservations {
276      all r:Reservation| {
277          r.reservor.isRegistered[]
278          not r.reservor.isBanned[]
279      }
280  }
281  fact userCannotReserveManyCarsInTheSameArea {
282      no disjoint r1,r2:Reservation| {
283          r1.reservor = r2.reservor
284          r1.car.area = r2.car.area
285      }
286  }
287  fact sameCarReservationIsNotPossible {
288      no r1,r2:Reservation| r1.car = r2.car
289  }
290  /*
291      The elapsedMinutes relation is a simplification of the creationTime field in the
     class diagram.
292      The elapsedMinutes is the only relevant information for the system as it is, and
     can be computed
293      quite easly from the creationTime.
294      The beforeGettingBanned expresses a temporal constraint, the system cannot prevent
     a user from
295      riding in the following situation:
296          a user reserves two cars,
```
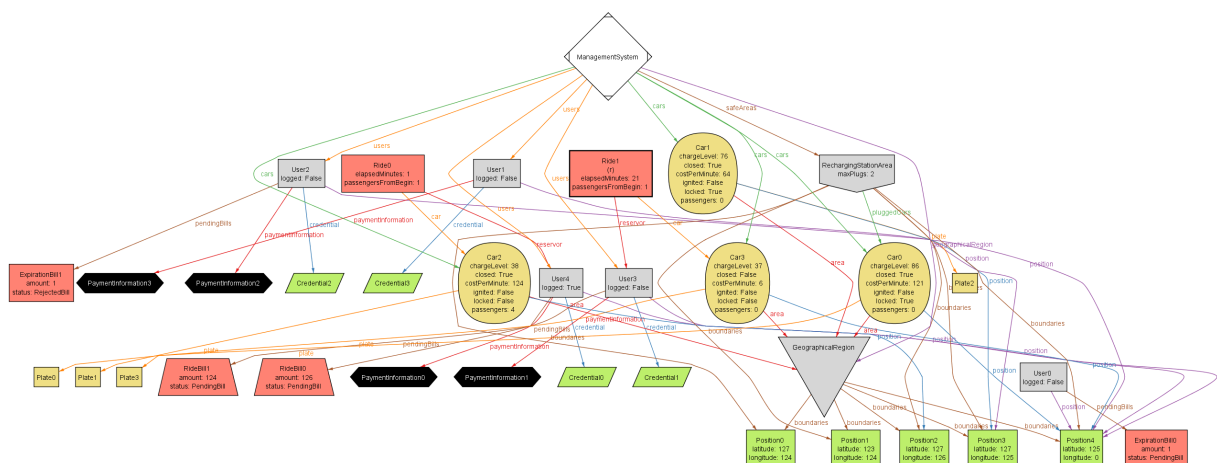
```alloy
297            then he starts a ride in one of the two,
298            the other reservation expires and the user is banned because he cannot pay
299    */
300    sig Ride {
301        car: one Car,
302        reservor: one User,
303        elapsedMinutes: one Int,
304        passengersFromBegin: one Int,
305        private beforeGettingBanned: one Boolean
306    } {
307        elapsedMinutes >= 0
308        passengersFromBegin >= 0
309        car.ignited=True => passengersFromBegin <= car.passengers
310    }
311    fact onlyRegisteredNotBannedUserCanHaveARide {
312        all r:Ride| {
313            r.reservor.isRegistered[]
314            r.beforeGettingBanned = False => not r.reservor.isBanned[]
315        }
316    }
317    fact userCannotRideManyCarsAtTime {
318        no disjoint r1,r2:Ride| {
319            r1.reservor = r2.reservor
320        }
321    }
322    fact onlyUnlockedCarsHaveARideAssociated {
323        all c:Car| {
324            c.locked = False <=> one r:Ride| r.car = c
325            c.locked = True => no r:Ride| r.car = c
326        }
327    }
328    pred show(){
329        #(RechargingStationArea.pluggedCars) > 0
330        some r:Ride| r.elapsedMinutes>0
331        (#Bill) > 0
332    }
333    run show for 5 but 8 Int
334    /*
335        This is a signature representing a tuple composed of
336            a unique driving licence number,
337            a unique username,
338            a password,
339            a date of birth,
340            a complete name
341    */
342    sig Credential {}
343    sig PaymentInformation {}
344    sig User {
345        paymentInformation: lone PaymentInformation,
346        credential: lone Credential,
347        logged: one Boolean,
348        position: one Position,
349        pendingBills: set Bill
350    } {
351        //An unregistered user cannot be banned or logged into the system
352        not this.isRegistered[] => {
353            logged = False
354            not this.isBanned[]
355        }
356        //all registered user belongs to the system, the other are free
357        this.isRegistered[] <=> credential != none
358        this.isRegistered[] <=> paymentInformation != none
359    }
360    pred User.isBanned[] {
361        some b:this.pendingBills| b.status = RejectedBill
362    }
363    pred User.isRegistered[] {
364        this in ManagementSystem.users
365    }
```

```
366    fact credentialAreAllUsed {
367        User.credential = Credential
368    }
369    fact credentialsAreUniqueByUser {
370        no disjoint u1,u2:User | {
371            u1.isRegistered[]
372            u2.isRegistered[]
373            u1.credential = u2.credential
374        }
375    }
376    fact paymentInfoAreAllUsed {
377        User.paymentInformation = PaymentInformation
378    }
379    fact paymentInfoAreUniqueByUser {
380        no disjoint u1,u2:User | {
381            u1.isRegistered[]
382            u2.isRegistered[]
383            u1.paymentInformation = u2.paymentInformation
384        }
385    }
```

World example

Appendix

Software and tools used

- Github (https://github.com) for version control

- Alloy (http://http://alloy.mit.edu/alloy/) for model verification

- StarUML (http://staruml.io/) for UML diagrams (except use case diagram)

- Signavio (http://www.signavio.com/) for use case diagrams

- Sublime Text Editor (https://www.sublimetext.com/) for alloy sintax highlighting

- Dillinger (http://dillinger.io/) as markdown editor and conversion from MarkDown to HTML

Hours of work

- Nardo Loris: 42 hours

- Osio Alberto: 38 hours