# Requirements Analysis and Specification Document


November 13, 2016

Giacomo Bossi

Marco Nanni

# Content

# 1. INTRODUCTION

## 1.1 Purpose

We will design and implement PowerEnJoy, which is a digital management system for a car sharing service that exclusively employs electric cars. It is based on a mobile application that users can access by providing their credentials.

The service has to provide the functionality normally provided by car-sharing services, like permitting users to search all the available cars within a certain distance from his or her localization (provided by GPS) or in a specific area, choosing one car and picking it up.

Clients need to be registered since only those people who own a valid driving license can access the service. The system also wants to store information about who is driving the car and keep track of all user's rides.

## 1.2 Scope

The system that mostly matches the characteristics of the infrastructure and the service we will provide is an application developed for the most common used mobile operating systems like IOS, Android and Windows Mobile. In order to interact with the application, the user should have a smartphone with an internet connection and an active GPS position sensor. The mobile application communicates with the system main database by providing online information about users' positions and statuses.

### 1.2.1 Domain Properties

We suppose that these conditions hold in the analysed world:

- Every electric car's position is provided by a GPS sensor installed inside the car
- When a user drives a car, its batteries discharges
- A car can be charged in specific power grid stations located in different areas of the city
- A user reserves a car only if he/she needs to use it
- A user can drive only one car at a time
- Only users who own a valid driving license can drive PowerEnJoy cars

### 1.2.2 Goals

- [G1] A person who owns a smartphone with an Internet connection must be able to download the PowerEnJoy application for free
- [G2] People who have downloaded the app must be able to sign up
- [G3] People who have already signed up must be able to log in
- [G4] Users must be able to modify their personal information
- [G5] Users must be able to search all the available cars within a certain distance from their current location or from a specified address
- [G6] Users must be able to get car's battery level before reserve the car
- [G7] Users must be able to reserve a car
- [G8] Users must be able to find the car they have reserved
- [G9] Users must be able to enter the car they have reserved
- [G10] Users must be able to take passengers
- [G11] Users must be able to drive the car they have reserved
- [G12] Users must be notified of their current charges during their rides
- [G13] Users must be able to recharge cars' batteries
- [G14] Users must be able to park their rented car in a safe area
- [G15] Users must be able to end their ride
- [G16] Users must be able to have discounts applied on their rides
- [G17] Users must be able to call a customer service if they have a problem

### 1.2.3 Domain Assumptions

We have to assume some facts in order to clarify some uncertain points provided in the specification document:

- Users must have a valid driving license
- Users must have a valid credit card
- Users must have a mobile phone with an Internet connection and with a GPS sensor
- Every car is equipped with a GPS sensor in order to recognise its own position
- Every car is equipped with a sensor that detects the number of passengers inside the car
- Every car has a battery charge level sensor
- Every car has internal sensors that detect the mechanical conditions of the vehicle
- Every car also has an internal computer that sends messages and information about sensors' detections to the system via an Internet communication
- Safe areas in which every user must park his/her car are recognizable on the map displayed on the screen installed inside the car
- Users can go outside safe areas but they cannot park rented cars there.

## 1.3 Definitions, Acronyms and Abbreviations

### 1.3.1 Definitions

- **Area**: city zone delimitated by at least three positions on the map
- **Safe areas**: areas in the city of Milan pre-defined by PowerEnJoy management system where users must park their rented cars
- **Border areas**: areas in the city of Milan pre-defined by PowerEnJoy management system where users can drive their rented cars but where they cannot park them
- **Power grid station**: a parking area where there are charging points
- **Charging points**: points where users can plug their rented cars in order to recharge them
- **Available car**: a car is available if no user has reserved it less than one hour ago and no user is driving the car
- **Reservation**: a user can reserve an available car for up to one hour without an additional fee. All the other users cannot see the car on the map as available anymore
- **Passenger**: a user can take passengers for the ride. Passengers do not need to be registered to the PowerEnJoy application.
- **Proximity**: a user is considered near a PowerEnJoy car if the user's position (provided by his/her GPS sensor) is at a maximum of 10 meters from the user's reserved car

### 1.3.2 Acronyms

- GPS: Global Positioning System

### 1.3.3 Abbreviations

- [G]: Goal
- [RE]: Requirement
- ID: Identification Card
- SSN: Social Security Number

### 1.3.4 Synonyms

- Area = Zone
- Available Car = Free Car
- Log in = Sign in

## 1.4 Identifying stakeholders

PowerEnJoy stakeholders are:

- Green Energy Company which is the main company that has launched PowerEnJoy
- Electric Car Company- which owns a part of PowerEnJoy actions - that has produced all the electric cars given to this new ecological car sharing service
- The Italian government

In fact, PowerEnJoy had been able to receive incentives from the Italian State because it is the first car sharing service entirely based on electric cars. The scope of these incentives is to reduce pollution and $CO_2$ emissions in the area of Milan.

Moreover, the system could be spread in other cities in the future.

## 1.5 Actors

- **Visitor user**: all visitor users can download the PowerEnJoy mobile application from the app market of their smartphones and visit the sign up page. They can complete the registration by providing their personal information (such as name, surname, age, driving license number, ID number, …) to gain access to all the application's functionalities.

- **Registered user**: once logged in, users can access all the functionalities provided by the application, such as search for an available car, reserve a car and pick it up, update their profile information.

## 1.6 Reference Documents

1. IEEE Recommended Practice for Software Requirements Specifications: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5841
1. Alloy model file: PowerEnjoy.als
2. Examples from past year: RASD sample from Oct. 20 lecture.pdf

## 1.7 Overview

This document is structured in seven sections:

- Section 1: Introduction
  It gives a general description of the system that we're going to design and implement, through a clarification of the goals, stakeholders and actors.
- Section 2: Overall Description
  This part gives general information about the product perspectives and functionalities with a focus about constraints, dependencies and assumptions
- Section 3: Requirements
  In this section we list all the functional and non-functional requirements
- Section 4: Scenarios
  This part is composed by a miscellaneous of situation that we have foreseen in order to give a simpler vision of the actors involved and possible employments of the system
- Section 5: Models
  In this part there are Use Case Diagrams, Use Case Diagrams Descriptions supported by Mockups, Sequence Diagrams, Class Diagrams, State Diagrams and Activity Diagrams
- Section 6: Alloy
  In this section we support the previous diagrams with an Alloy analysis
- Section 7: Appendix
  The Appendix is composed by a reference of software and tool used and the working hours each teammate spent for the accomplishment of the RASD

# 2. OVERALL DESCRIPTION

In this part of the document we will focus on a general description of what we are going to implement.

## 2.1 Product Perspective

The infrastructure that we are presenting is composed by three main parts:

- Mobile Application
- Additional Hardware on the car
- Central Server Architecture

### 2.1.1 Mobile Application

Users can download PowerEnJoy application from the app market and they can access the application wherever they are with their smartphone. An Internet connection is required in order to provide an up to date map displaying all the available cars' positions.

Users can use the mobile application in order to interact with the car sharing digital management system. In this way, they can search all the available cars throughout the metropolitan city of Milan. In particular, they can search for an available car in a specific area by providing a specific address or by their localization, which is detected by the GPS sensors of users' smartphones. They can also reserve a car and thus the system allow users to have sufficient time to reach the car and pick the car up.

### 2.1.2 Additional Hardware on the car

PowerEnJoy electric cars distributed by Electric Car Company include a special equipment in order to provide all the functionalities of this new car sharing service. Every PowerEnJoy car is equipped with these tools:

- GPS sensor
- Internal HD RGB Camera
- 10" LCD Screen mounted on the central side of the dashboard

The GPS sensor sends the car's position to the system in order to always provide an up to date showing of all the available cars.

The internal HD RGB camera detects how many passengers are inside the car and sends the information to the system; according to the number of passengers, the system will apply a discount on the last user's ride.

The LCD screen mounted on the central side of the car's dashboard displays information about the user's ride. For example, it shows the user his/her position and destination, the user's path, the car's battery level, the nearest power grid station and its address, the user's current charge and if the user is in a safe area.

Each tool communicates with the central computer mounted inside the locked glove compartment of the car. The computer sends data streams to the central server architecture

### 2.1.3 Central Server Architecture

PowerEnJoy requires a database in order to store all data coming from cars' sensors; in fact, the system has been developed in order to keep track of all users' rides and paths.

## 2.2 Product Functions

PowerEnJoy mobile application will provide these main functionalities:

- Allow users to search all the available cars by showing cars' positions in a map of the city of Milan
- Show the cars' battery levels
- Show the cars' distances from users
- Allow users to rent a car
- Allow users to recharge their rented cars' batteries

The car's software will provide these main functionalities:

- Allow users to enter the rented car
- Allow user to start using the rented car
- Show the user his/her current charge during the ride
- Show the user all the safe areas through a map displayed on the car's screen
- Show the user his/her current position on the map
- Allow users to stop using the rented car

## 2.3 User characteristic

People that will use our mobile application are adults (more than 18 years old) owning a regular driving license. Users must be able to download and access PowerEnJoy application by their smartphone with an Internet connection. The application is available for free in the app market and users can log in by providing their personal information.

## 2.4 Constraints

### 2.4.1 Regulatory policies

Due to the fact that PowerEnJoy is the first car service entirely based on electric cars, it has received incentives from the Italian State and so it has to report periodically to the Italian government data and statistics about their cars' employments. In this way, the government wants to be informed about their results in order to decide or not to export this service in other cities throughout Italy.

### 2.4.2 Hardware limitations

The mobile app needs the following hardware specification to be installed:

| Mobile OS Version | Available Memory | RAM | GPS | Mobile Connection |
|---|---|---|---|---|
| **Android 4.4 +** | ~50 MB | 1 GB + | yes | yes |
| **IOS 9.0 +** | ~40 MB | 1 GB + | yes | yes |
| **Windows Mobile 8.1 +** | ~40 MB | 1 GB + | yes | yes |

### 2.4.3 Interfaces to other applications

PowerEnJoy has a payments gateway that allows users to proceed with their registration only if they have a valid credit card. The payments gateway also handles every payment as soon as a user's ride is ended.

### 2.4.4 Parallel operations

PowerEnJoy has to support parallel operations from different users since it has to communicate with the database all the cars' statuses changes. This in order to always have an up to date map showing the instantaneous cars' availability.

## 2.5 Assumptions and dependencies

- The payments gateway will not allow users to proceed with their registrations if they do not have a valid credit card
- Every person can have at most one account. Since during the registration phase the user has to provide his/her driving license, more accounts based on the same driving license are not allowed
- If a user deletes his/her own account, he/she can re-create an account through the same procedure
- If a user reserves a car, that car is no more available and visible by the application from the other users
- When a user parks the car in a safe area and ends his/her ride and, the system allows the user to exit the car
- When the user's charge is stopped and the user exits the vehicle, the system waits for a prefixed period of time before locking the car
- After a car is locked, the system sets the car as available and makes it visible by all the users
- If the user detects a car's problem (inside or outside), the user has to communicate the problem to the customer service
- In case of problems or accident, users can call the customer service; operators are 24/7 available to users

# 3. REQUIREMENTS

## 3.1 Functional Requirements

Visitor:

- [RE1] sign up

User:

- [RE2] log in
- [RE3] modify his/her profile information
- [RE4] search for an available electric car
- [RE5] get information about available cars
- [RE6] reserve a car
- [RE7] delete a car's reservation
- [RE8] pick the reserved car up
- [RE9] tell the system he/she is near the car (proximity)
- [RE10] park the car in a safe area
- [RE11] recharge car's batteries in a special parking area
- [RE12] pay for his/her ride
- [RE13] Have discounts applied because of his/her virtuous behaviour

## 3.2 Non-Functional Requirements

The system will provide these main user visible aspects not directly related to the system functional behavior:

Performance

All the functionalities provided by the system are executed in an acceptable period of time and they do not cause any system block or process failure. As stated in different parts of the document, the user needs to have a sufficient Internet connection on his/her smartphone in order to see an up to date map showing all the PowerEnJoy available cars of the city.

Accessibility

PowerEnJoy app is accessible by every kind of user that have a smartphone and can download an application from the app market.

Availability

PowerEnJoy app can be downloaded for free from the app market of a smartphone and the service is 24/7 available to users.

Portability

PowerEnJoy application is developed for all the main mobile operating systems now available on the market like Android, iOS and Windows Phone.

Concurrency

The system guarantees data consistency in the database which contains all the data coming from both users and cars.

Security

In order to guarantee the security, the system allows visitors of the app to only fill the registration form. All the other functionalities provided by the system require the user to log in.

Moreover, to provide a higher security level, the system will make the car be started only if the user enters correctly his/her pin code.

# 4. SCENARIOS

This section will present some possible situations that might occur by the interaction between a guest or a user and the developed PowerEnJoy application.

### 4.1 Mario registers to PowerEnJoy

Mario have heard about this new ecological car-sharing service and given that he often uses car-sharing services to go to work, he decides to try the PowerEnJoy service. He downloads PowerEnJoy application from the app market by his smartphone and starts the registration procedure.

He has to fill the pre-defined form by entering name, surname, sex, date and place of birth, username that he wants to use, phone number, email, identification card number, its date of issue and its expiration date, social security number, driving license number, a credit card number. All these fields are mandatory.

Afterwards, if all the filled fields are correct and if the payments gateway has checked the validity of his credit card provided, Mario will receive an email confirmation containing his username, a system generated password and a pin code that he has to provide whenever he will enter a PowerEnJoy car. With username and password, he can log in the application and enjoy all the PowerEnJoy functionalities.

### 4.2 Giacomo searches for the nearest PowerEnJoy car around him

Giacomo, who is already a PowerEnJoy user, has just left work and one of his colleagues has told him that today afternoon there is a public transport strike. No one in his family can help him and so decides to search on his smartphone if there is an available PowerEnJoy car around him.

He knows that when there are emergencies like strikes, it is more difficult to find a car in the neighbourhood. Moreover, his house is far away from his office and so he needs a car with a sufficient battery charge level. From the map displayed on the phone's screen, he can see all the available cars, their distances and their battery level. He finds a car distant 2 km from him with 80% of battery level and so he decides to reserve it. Now he has 1 hour to reach the car before an additional fee is applied from the system and the reservation expires.

### 4.3 John searches for a car in a specific position

John is a foreign student at Politecnico who came in Milan to attend the first year of Master degree. He's already a PowerEnJoy user and tomorrow night he will hang out with his friends in the centre of Milan. He does not have his own car here in Milan and he doesn't know how to come home. His friends told him that ATM service (which is the public transport service in Milan) is good but after midnight there are not so many buses as during the day and the subway is closed.

So John is thinking that if there are several PowerEnJoy cars near Duomo now, probably there will be approximately the same number of cars even tomorrow night so that he can take one and come back home with a PowerEnJoy car.

He decides to search on the map of the mobile application all the available cars located near the Duomo by entering “Duomo cathedral” in the search bar. There are lots of PowerEnJoy cars in the centre of Milan now and so probably there will be one car tomorrow night too.

In the end of the evening, two friends of John do not know how to come back home, like Giacomo the day before. So Giacomo decides to take them onto his PowerEnJoy rented car and discovers that the system will apply a discount of 10% on his ride.

### 4.4 Monica recharges her rented PowerEnJoy car's batteries

Monica does not have her own car because it is expensive and she also prefers to use public transports in order to reduce $CO_2$ emissions that are causing a problem in the metropolitan city of Milan, which is always traffic congested. But, sometimes, she would like to use a car because it is more comfortable, especially when she has to come back from the supermarket with lots of heavy bags.

As soon as she has heard about this new ecological car-sharing service, she signed up and tried PowerEnJoy cars. Since PowerEnJoy built a safe area near the supermarket she is used to go to, she started to use more and more often this new service.

Moreover, the safe area where Monica parks her car, gives the opportunity to recharge car's batteries. When Monica takes care of plugging the car into the power grid, the system applies a discount of 30% on her last ride.

### 4.5 Marco called the customer service

Marco is a PowerEnJoy user and he uses these ecological cars every day to go to the university. But today, while he was driving a PowerEnJoy car, he saw from the screen mounted inside the car that his current charging was unusual. As soon as he has parked the car, he called the customer service and explained his problem. The customer service, which is 24/7 available to PowerEnJoy users, started immediately a charge recovery procedure in order to give money back to Marco.

### 4.6 Judy goes to a concert

Judy loves music and now that she has moved to Milan to study at the university, she often goes to Assago Milano Forum to listen some music bands. Today she is going to listen her favourite pop band; she decides to go with a PowerEnJoy car but she doesn't know if cars can be parked there. She accesses the application in order to understand if she can do it and she finds out that there is a big parking area reserved for PowerEnJoy cars where users can also recharge their cars' batteries. So she reserves an electric car, she reaches the car and then she starts driving it.

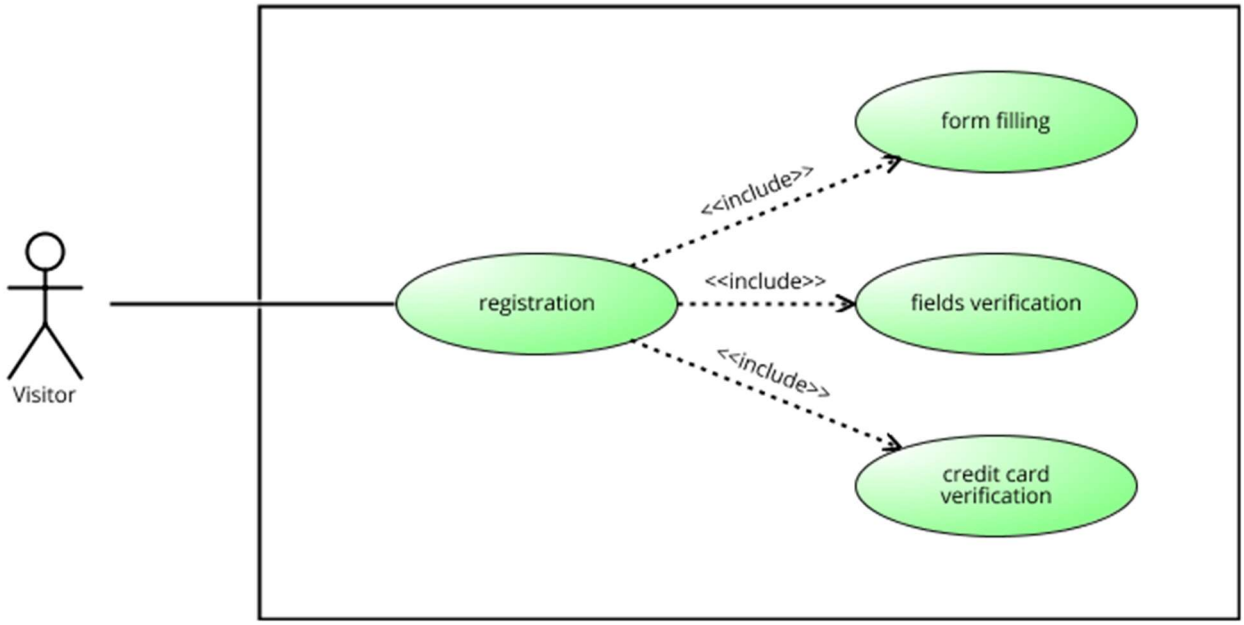
Once she reaches the stadium, she realizes that lots of people have picked a PowerEnJoy car up and there are no more parking spaces; so she starts going around to find a place where she can park her car. Later she finally finds a parking space where she can leave her rented car but since the car is more than 3 km away from the nearest power grid station (which is the one inside the PowerEnJoy parking area near the stadium), she has to pay an additional charge on her ride.

# 5. UML MODELS

In this section we are presenting an abstraction of the main functionalities provided by our system and different interactions between visitors or users and PowerEnJoy application.

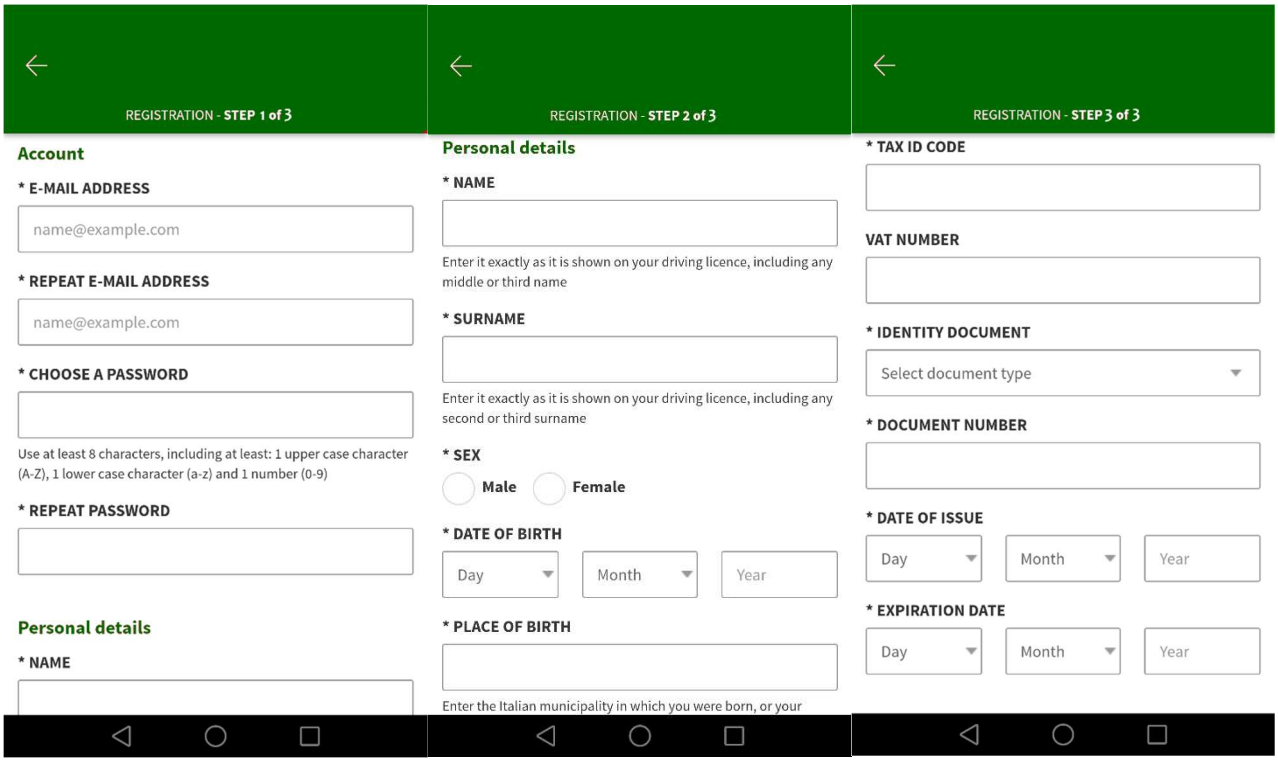
## 5.1 Use case diagrams and descriptions

UC.1 - Registration



5.1.1 Registration

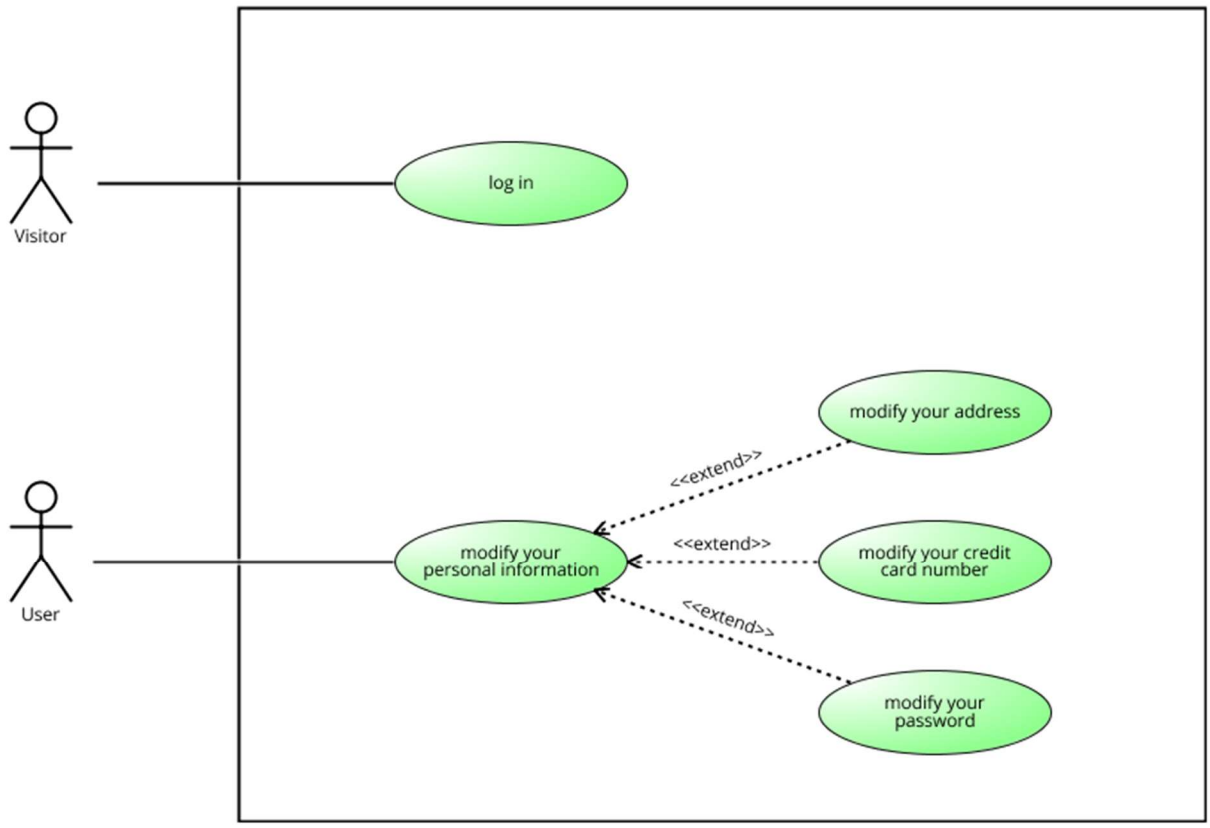
| **Actor** | Visitor |
|---|---|
| **Input condition** | The visitor is a person who has downloaded the PowerEnJoy application from the app market on his/her smartphone and has not already signed up. |
| **Events Flow** | The visitor, who wants to become a user of the application, fills the pre-defined form by compiling all the mandatory fields. |
| **Output condition** | If all the mandatory fields have been filled correctly and if the payments gateway has verified the validity of the credit card provided, the system adds the new user to the database. The system also sends the user an email containing the username provided by the user during the form compilation, a generated password associated to that username and a pin code that the user has to provide when he/she enters the car. The user must use these username and password in order to log in the application. |
| **Exceptions** | If at least one field is missing or if it is wrong, a warning message is shown and the user can re-insert the missing fields. |



5.1.2 Log in

| **Actor** | User |
|---|---|
| **Input condition** | The user is a person who has already signed up |
| **Events Flow** | The user can access the PowerEnJoy application by providing the username that he/she has chosen during the registration phase and the password provided by the system. |
| **Output condition** | The user can explore and enjoy all the functionalities of the application. |
| **Exception** | If username and password are not correct, the user has to reinsert them. |

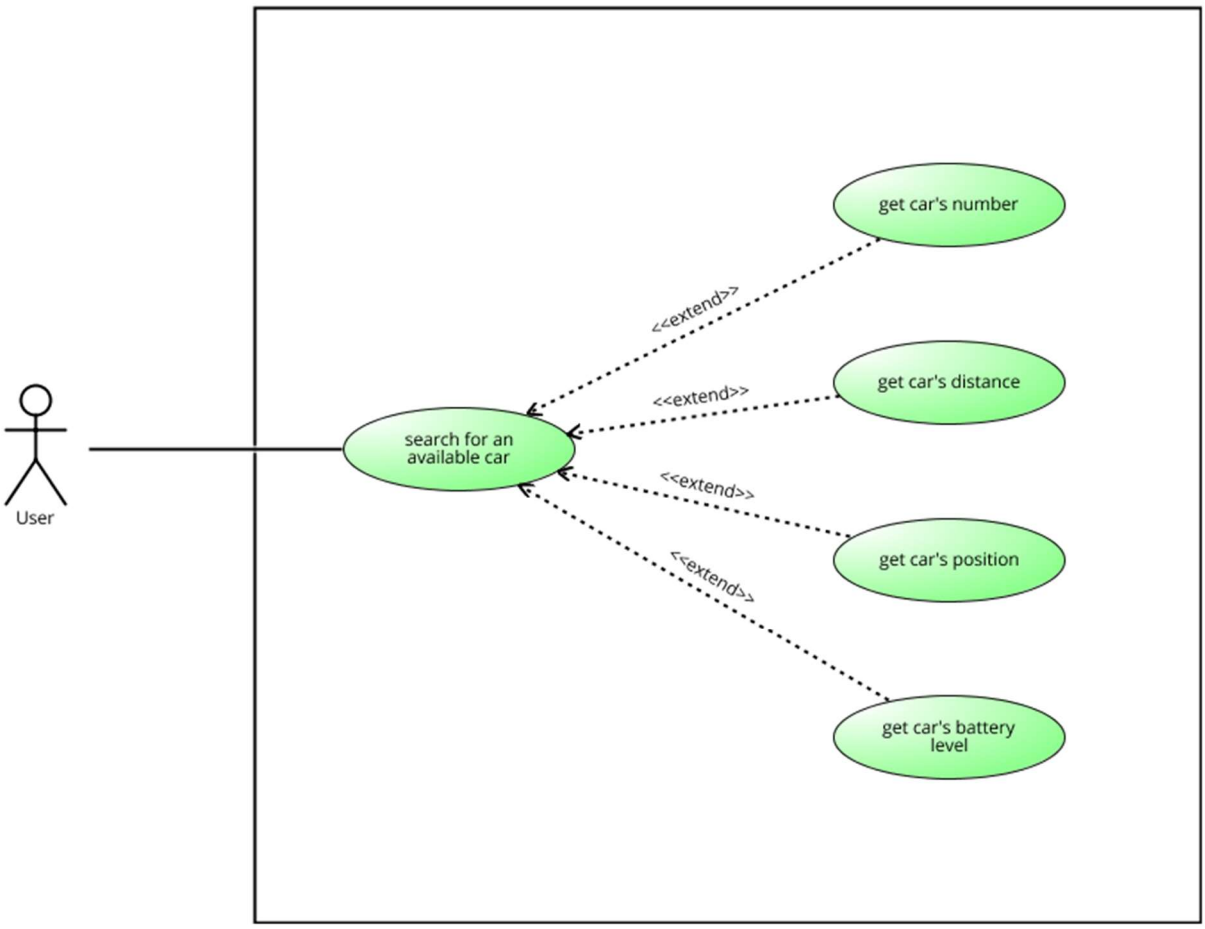
UC.2 – Modify your personal information




5.1.3 Modify your personal information

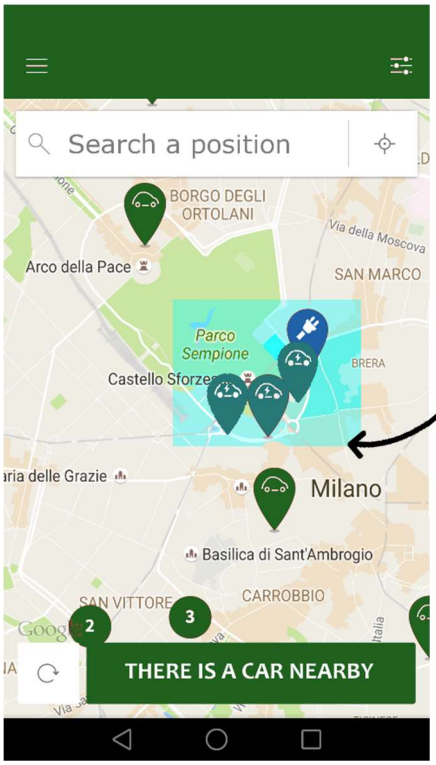
| **Actor** | User |
|---|---|
| **Input condition** | The user has already signed up |
| **Events Flow** | Once logged in, the user can modify his/her personal information in the profile *page* of the application such as the address (e.g. the user could have changed his/her residence), the credit card associated to the user, the user's password. |
| **Output condition** | If all the modified fields are correct and the payments gateway has verified the validity of the credit card, the new filled fields are updated. |
| **Exception** | If at least one field is not correct, the user has to recompile that field. |

UC.3 Search for an available electric car and get information about it



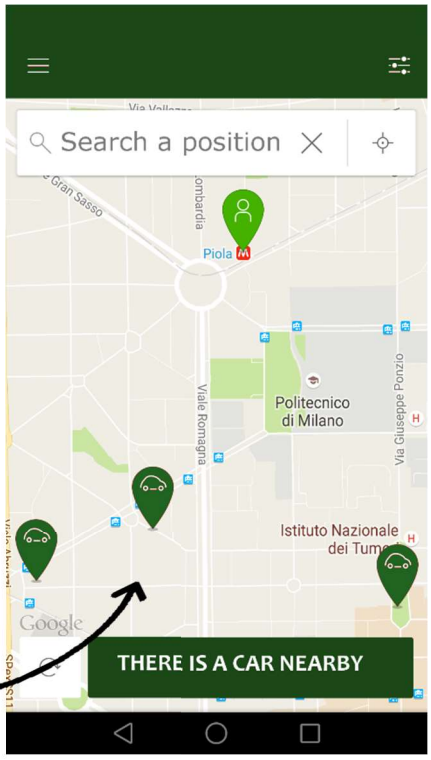
5.1.4 Search for an available car

| **Actor** | User |
|---|---|
| **Input condition** | The user has already signed up |
| **Events Flow** | Once accessed the application, the user inserts a specific address in the search bar or clicks on the localization icon. This icon permits the user not to manually insert an address, because the system detects the user's position via the GPS sensor installed inside his/her smartphone. |
| **Output condition** | The application shows a map with all the available cars that are represented through different icons. |
| **Exception** | If the system cannot detect the user's position because there are some problems with the GPS sensor, the user has to manually insert his/her position in the search bar.<br>If the specific address inserted is not correct, an error message is shown. |



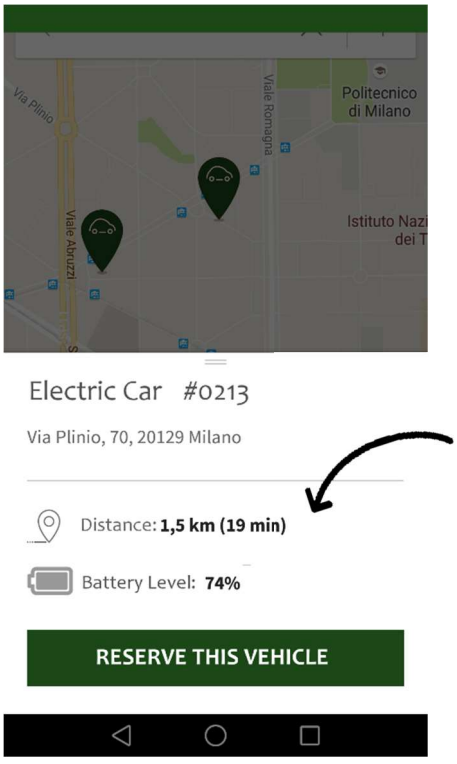

The light blue zone indicates a Power Grid Station. In this case there are four Charging Points, three of them are occupied by charging cars and one of them is free.

Here there is the user's position indicated by a light green icon. There are also some parked cars in the neighbourhood indicated with a darker green icon.



5.1.5 Get information about an available car

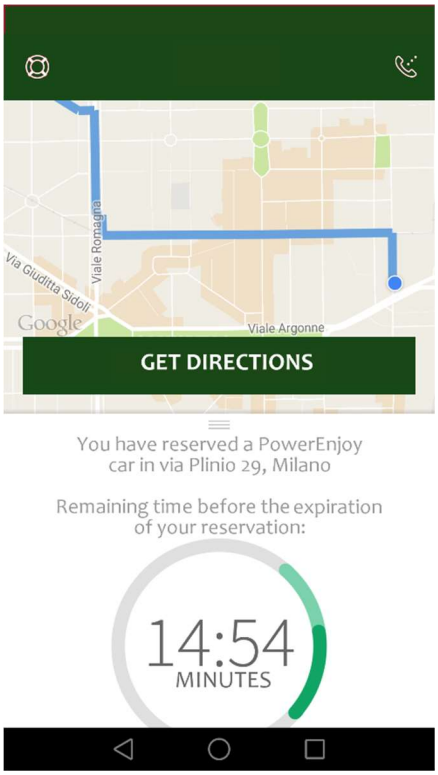
| **Actor** | User |
|---|---|
| **Input condition** | The user has already signed up and he/she has already searched for a car |
| **Events Flow** | On the available car details screen, the user taps a car's icon (which represents an available car) in order to see the car's battery level, the specific distance from the user or from the specified address and an estimated time to reach the car. |
| **Output condition** | A screen appears showing the car's position in the map, the car's ID number, the car's address, the car's distance from the user or from the specified address, an estimated time to reach the car, the car's battery level. |
| **Exception** | If the user taps outside the icons, he/she returns back to the map screen. |




In this screen referred to the previous Use Case Description, there are all the car's information that the user will see. In fact, there is the distance between the user's position (or from the specified address) and the car, an estimated walking time, the car's position, the car's ID number and the car's battery level.

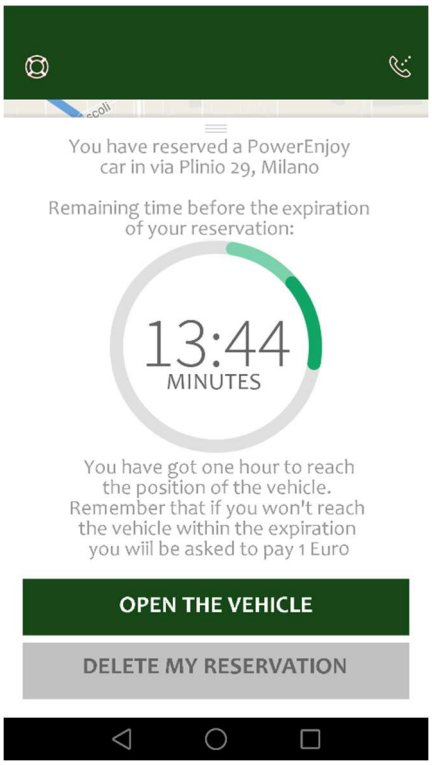
5.1.6 Reserve a car

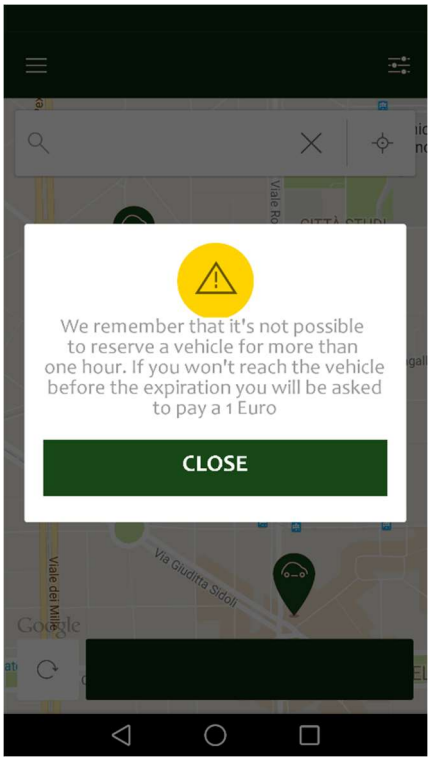
| **Actor** | User |
|---|---|
| **Input condition** | The user has already signed up |
| **Events Flow** | On the available car details screen, the user reserves a car by tapping the button “reserve this vehicle”. |
| **Output condition** | The car is reserved and no user can reserve that car. The user who has reserved the car has one hour to reach the car and pick it up. A screen will appear, showing the user how much time he/she has to reach the car and a possible path to find it. |
| **Exception** | If the user doesn’t reach the car within one hour from the reservation, the user pays a fee of 1 euro and the reservation expires. |



5.1.7 Delete a car reservation

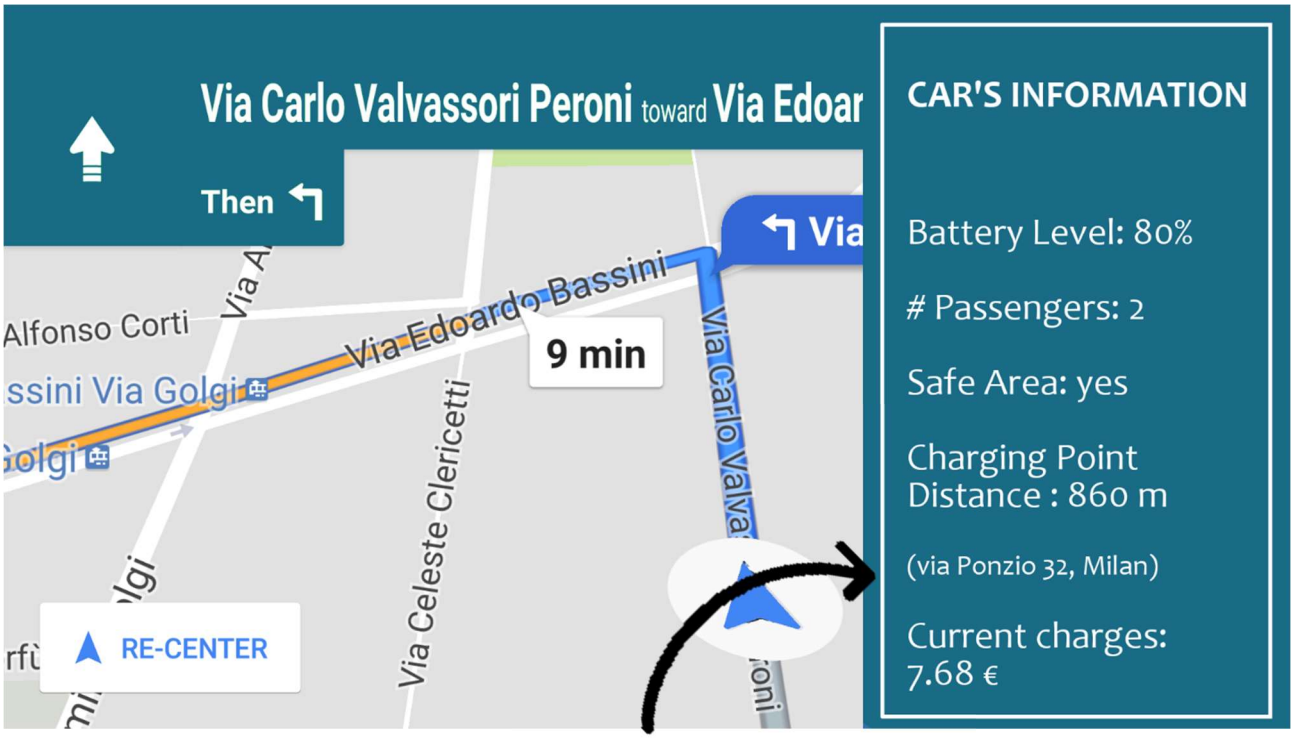
| **Actor** | User |
|---|---|
| **Input condition** | The user has already reserved a car |
| **Events Flow** | On the car reservation screen, the user taps the button “delete your reservation” in order to delete his/her reservation. |
| **Output condition** | The user can delete his/her reservation within 1 hour without having a fee applied.<br>Now the user can reserve other cars, except the ones he/she has already reserved. |
| **Exception** | The user can’t reserve the same car twice in 2 hours. If the user tries to do this, an error message appears. |



5.1.8 Pick the reserved car up

| **Actor** | User |
|---|---|
| **Input condition** | The user has already reserved a car |
| **Events Flow** | Once the user has reached the car, the user tells the system he/she is nearby and the system automatically unlocks the car. |
| **Output condition** | The user enters the car, pick the keys from the glove compartment, enter its personal pin code, start the car and drive the car. The system starts charging the user for a fixed charge per minute. A screen mounted inside the car displays the user's current charge. |
| **Exception** | If the user tells the system he/she is nearby but he/she does not enter the car in 2 minutes, the system locks the car and the user's reservation expires. |




This image is an example of what can be seen on the car' dashboard computer. It's possible to see some information about the battery level of the car, the number of passenger, the zone of the current position, the distance between the current position and the nearest charging point, the location of the nearest charging point and the current charge. Moreover, it's possible to use it as a navigator.

5.1.9 Park the car in a safe area

| **Actor** | User |
|---|---|
| **Input condition** | The user is driving a car |
| **Events Flow** | The display mounted inside the car shows a map of the city of Milan with all the safe areas highlighted. Users can park their cars only inside these areas.<br>Since he/she leaves the car, the system waits for 2 minutes and then locks the car automatically. |
| **Output condition** | The user parks his/her car inside a safe area and chooses to end the ride, so he/she exit the car. |
| **Exception** | If a user parks and exits the car outside a safe area, the system informs the user that he/she has 5 minutes to re-enter the car and park the car in a safe area. Otherwise an expensive fee will be applied to the user. |

5.1.10 Recharge car's batteries in a special parking area

| **Actor** | User |
|---|---|
| **Input condition** | The user has parked his/her rented car |
| **Events Flow** | Once the user has parked his/her car in a special parking area where it can be recharged, the user take care of plugging the car into the power grid. |
| **Output condition** | The user receives a discount of 30% on his/her last ride. |
| **Exception** | - |

5.1.11 Power grid discount

| **Actor** | System |
|---|---|
| **Input condition** | The user has plugged the car into a power grid |
| **Events Flow** | The user has parked the car in a special parking area and has taken care of plugging the car into a power grid in order to recharge it. In this way, the system applies a discount of 30% on the last user's ride to incentivize the virtuous behaviours of the users. |
| **Output condition** | The last user's ride payment is discounted of 30% |
| **Exception** | - |

5.1.12 Receive user charge's payment

| **Actor** | System |
|---|---|
| **Input condition** | The user has just end the ride |
| **Events Flow** | Once the user has end the ride and has exit the car, the system's payments gateway withdraws the user's ride charge from his/her credit card. |
| **Output condition** | The user pays for his/her last ride |
| **Exception** | If the credit card does not contain a sufficient amount of money to pay for the last ride, a warning message is sent to the user and he/she has *5 days* to pay. Otherwise an additional fee will be applied to the user. |

5.1.13 Apply discount for passengers picked up

| **Actor** | System |
|---|---|
| **Input condition** | The user has taken at least two other passengers onto the car during the ride |
| **Events Flow** | The system's sensors detect that there are at least 3 people (1 driver and 2 passengers) and applies a discount on the last user's ride. This is in order to incentivize the virtuous behaviours of the users. |
| **Output condition** | The last user's ride payment is discounted of 10% |
| **Exception** | - |

5.1.14 50% battery charge discount

| **Actor** | System |
|---|---|
| **Input condition** | The user has left the car with more than 50% of battery charge |
| **Events Flow** | Once the user has exit the car and has end his/her ride, the system detects that the car's battery level is greater than 50% and it applies a discount on the last user's ride. This is in order to incentivize the virtuous behaviours of the users. |
| **Output condition** | The last user's ride payment is discounted of 20% |
| **Exception** | - |

5.1.15 More charges for a car parked out of range

| **Actor** | System |
|---|---|
| **Input condition** | The user has parked his/her rented car at more than 3 KM from the nearest power grid station |
| **Events Flow** | The system detects that the position communicated by the car's GPS sensor is more than 3KM away from the nearest power grid station. The system charges 30% more on the last ride to the user because PowerEnJoy has to compensate for the cost required to recharge the car on-site. |
| **Output condition** | The last user's ride payment is increased of 30% |
| **Exception** | - |

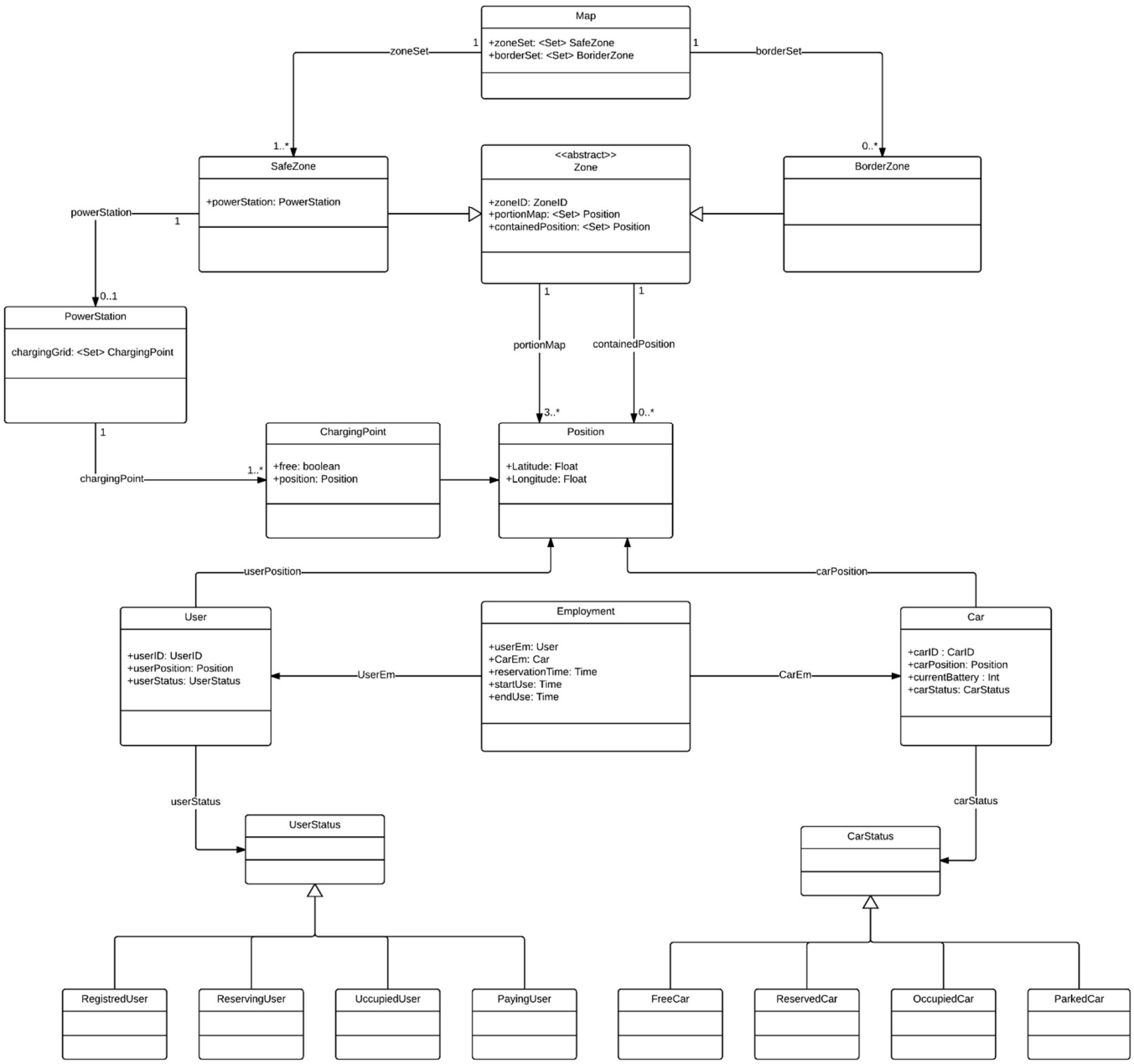
5.1.16 More charges for a car with more than 80% of the battery empty

| **Actor** | System |
|---|---|
| **Input condition** | The user has left his/her rented car with more than 80% of the battery empty |
| **Events Flow** | Once the user has exit the car and has end his/her ride, the system detects that the car's battery level is less than 20% and charges to the user 30% more on his/her last ride. |
| **Output condition** | The last user's ride payment is increased of 30% |
| **Exception** | - |

**Traceability matrix**

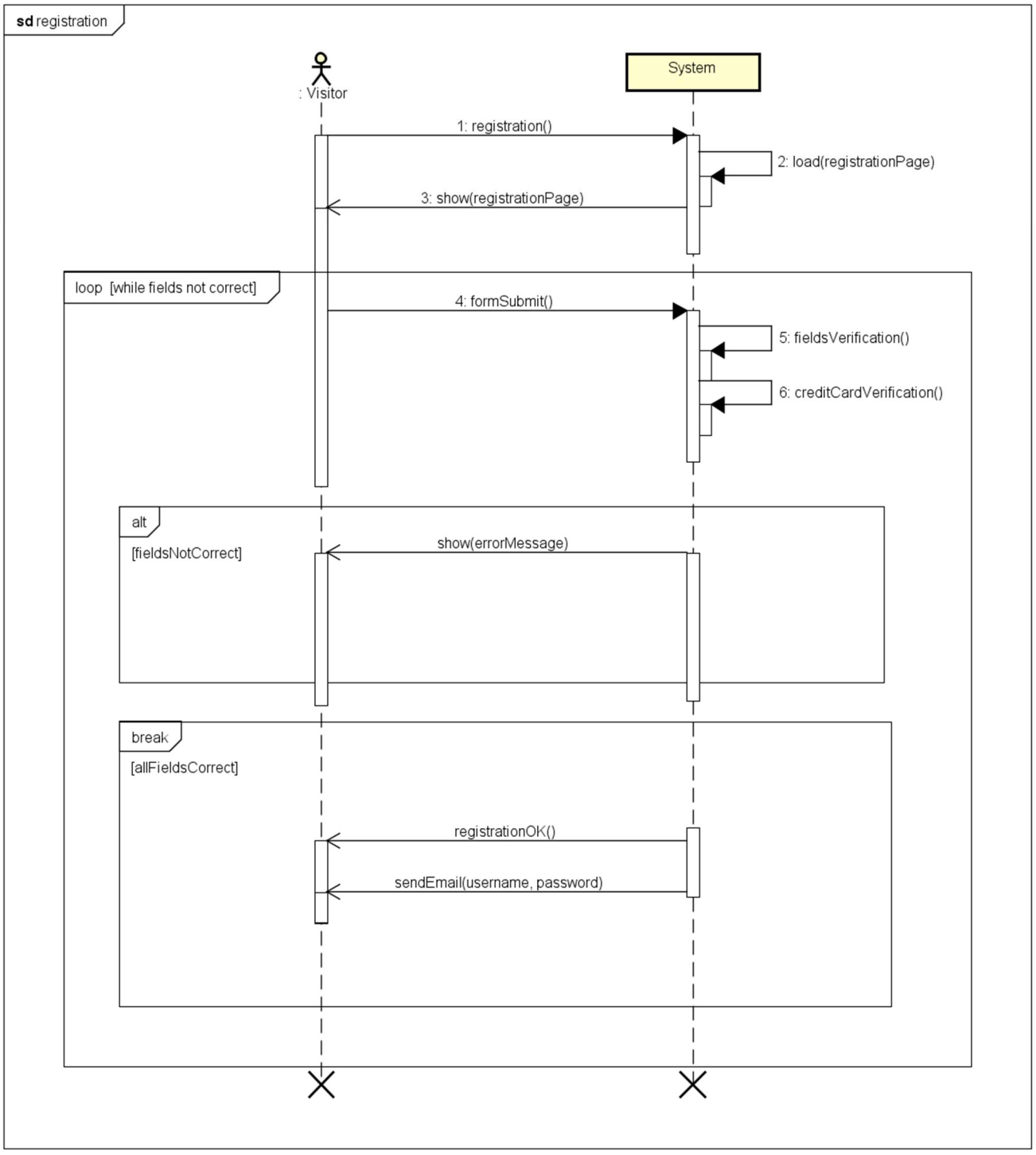
| Raw ID | Goal ID | Requirement ID | Use Case Diagram ID | Use case description ID |
|---|---|---|---|---|
| 1 | G2 | RE1 | UC.1 | 5.1.1 |
| 2 | G4 | RE3 | UC.2 | 5.1.3 |
| 3 | G5 | RE4 | UC.3 | 5.1.4 |
| 4 | G6 | RE5 | UC.3 | 5.1.5 |

## 5.2 Class diagram

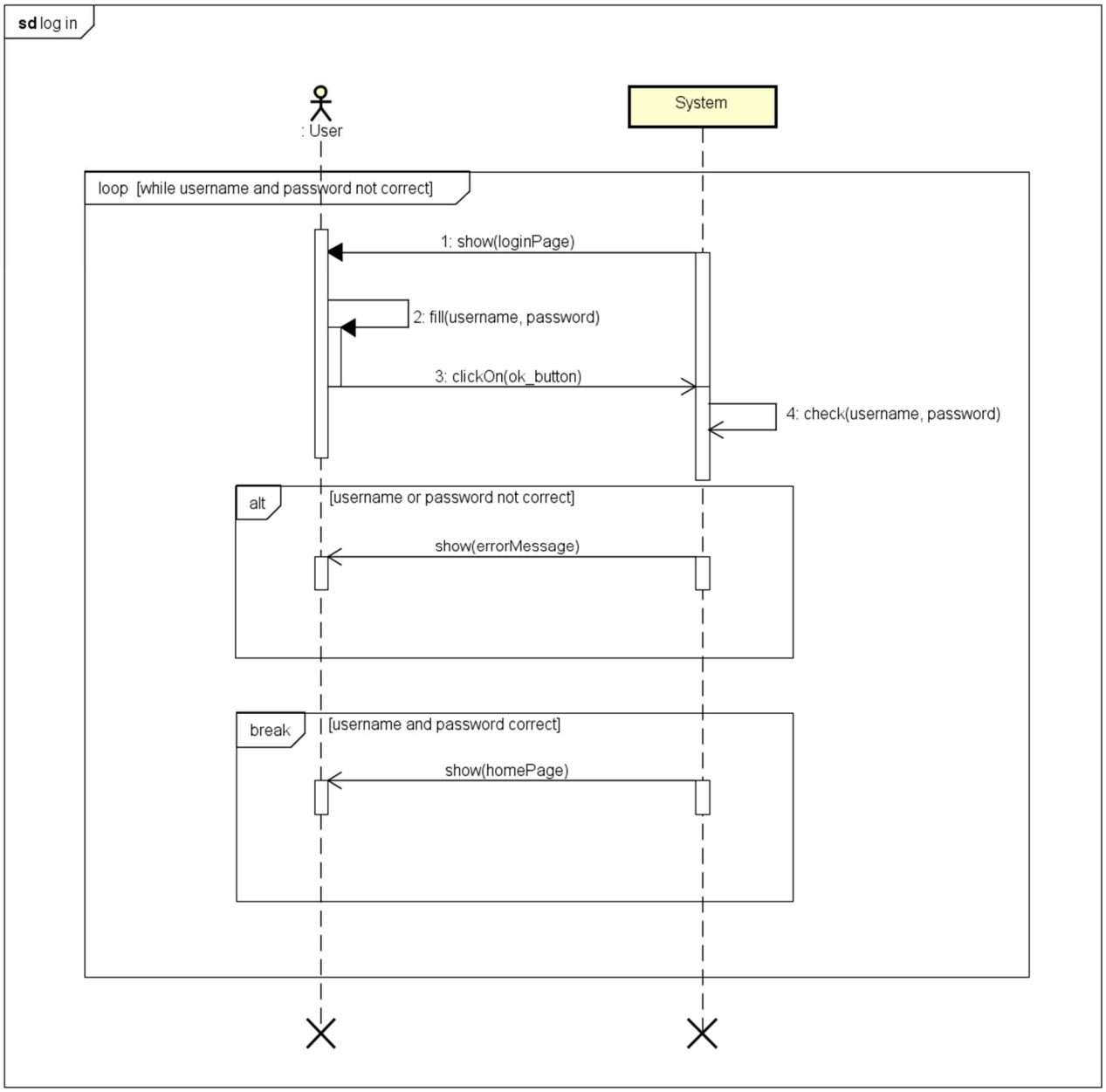


## 5.3 Sequence Diagrams

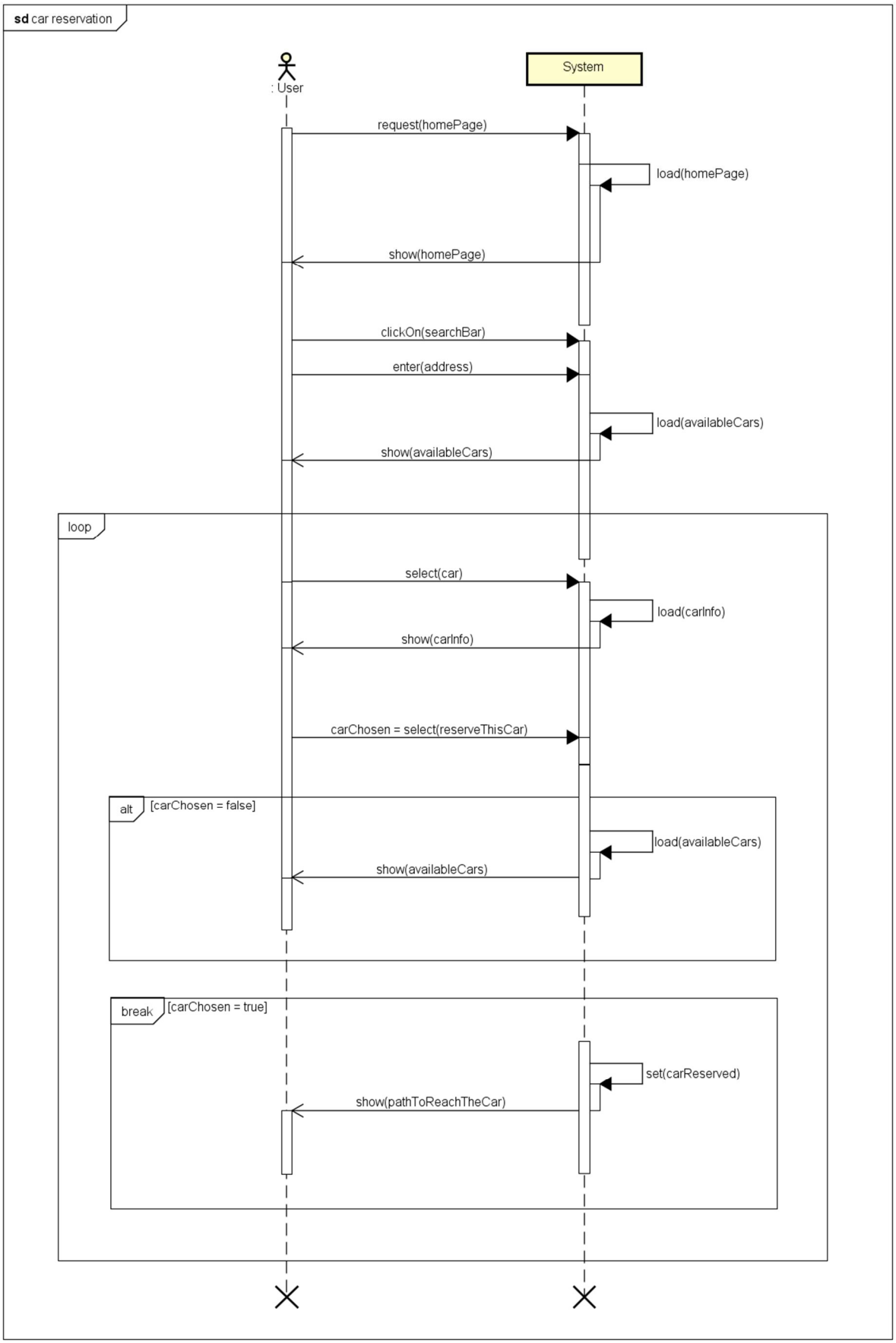
### 5.3.1 Registration



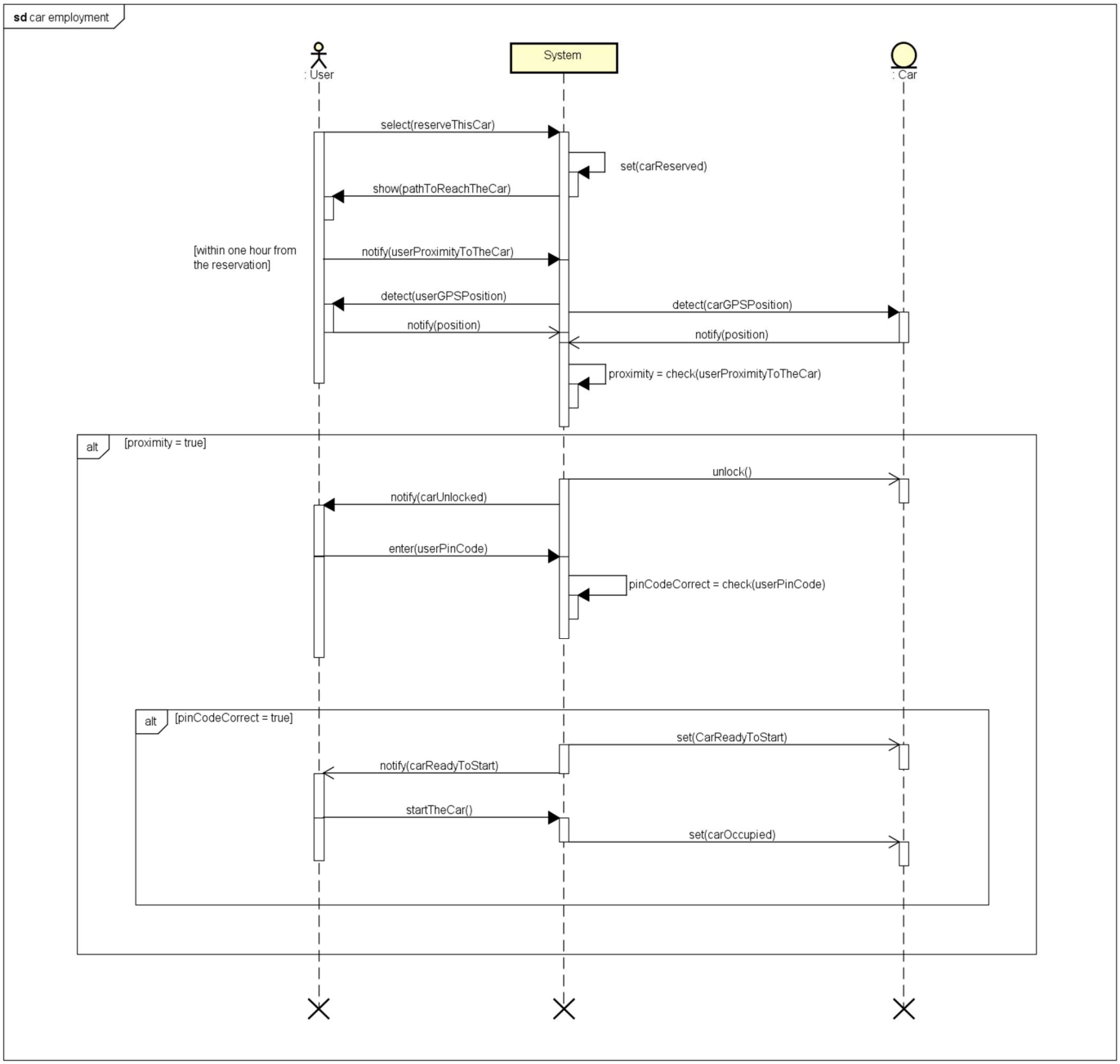
5.3.2 Log in


sd log in
: User
System
loop [while username and password not correct]
1: show(loginPage)
2: fill(username, password)
3: clickOn(ok_button)
4: check(username, password)
alt
[username or password not correct]
show(errorMessage)
break
[username and password correct]
show(homePage)

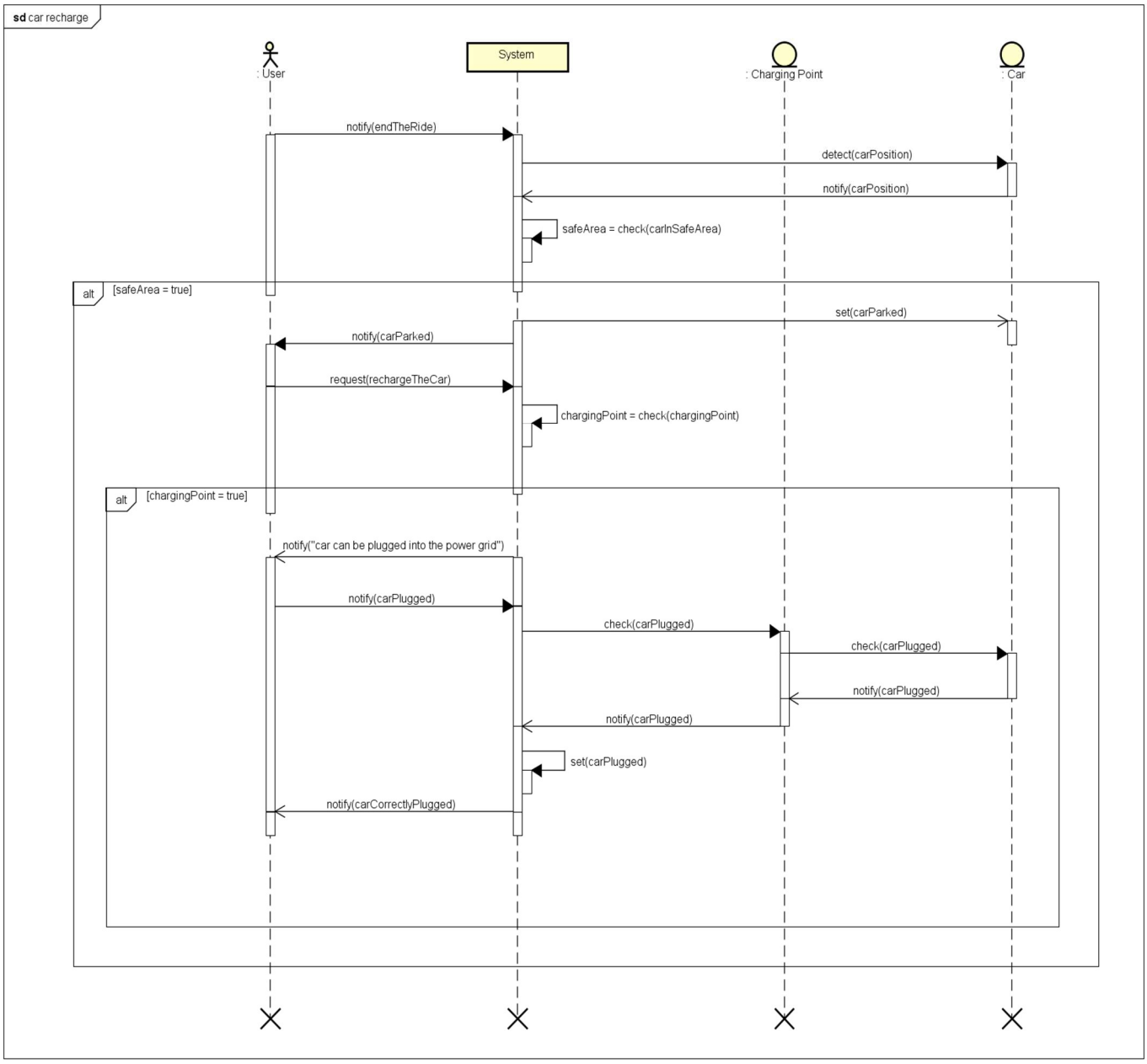
### 5.3.3 Car reservation



## 5.3.4 Car employment

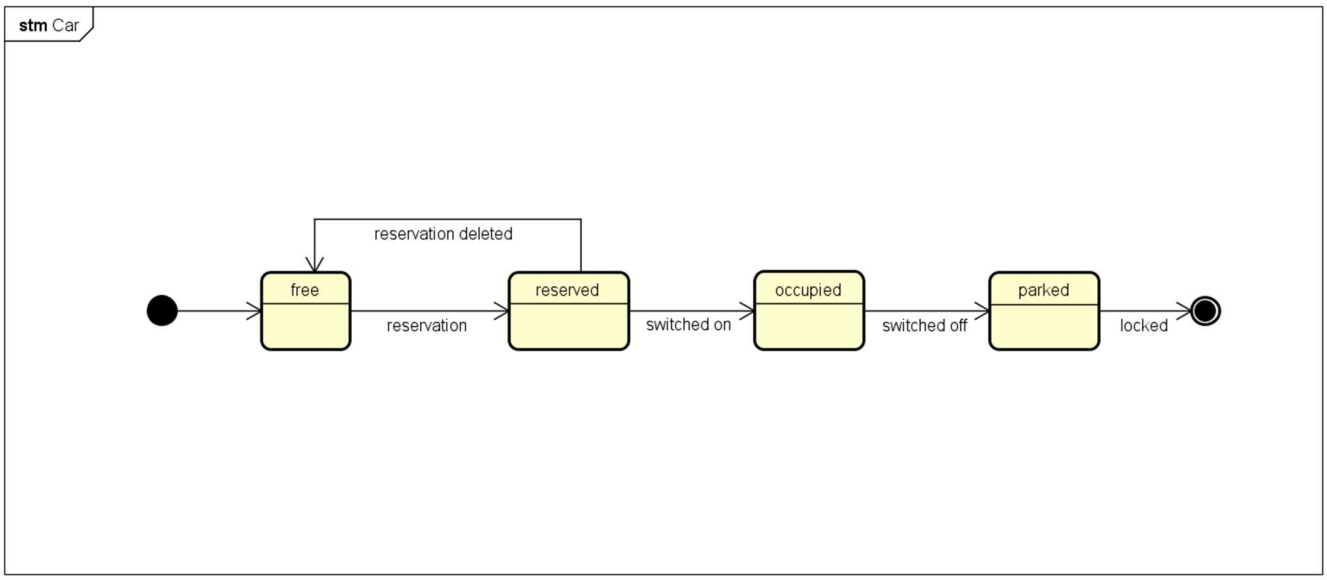


### 5.3.5 Car recharge

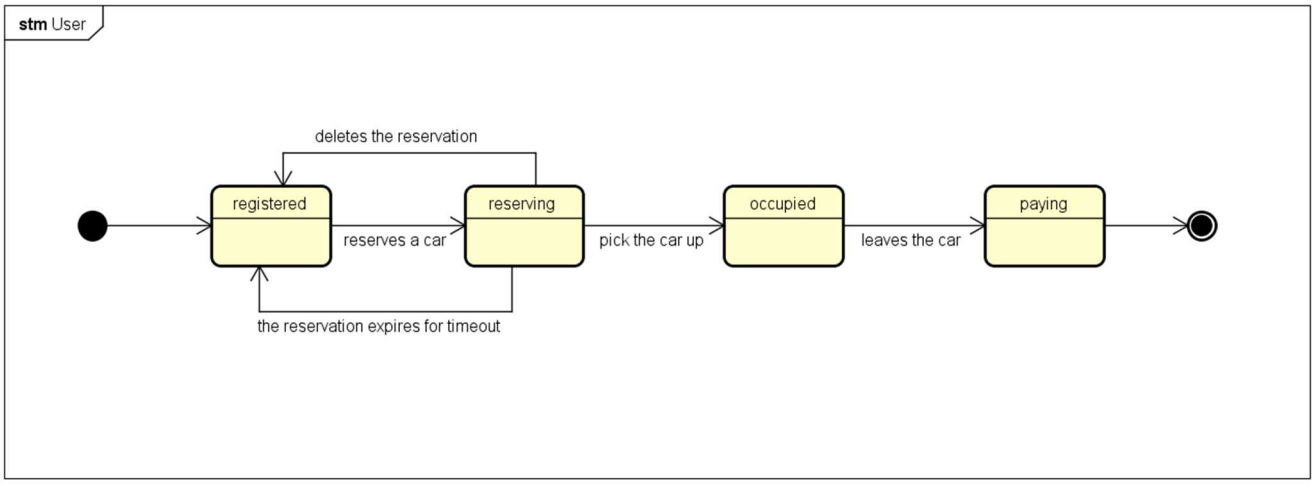


## 5.4 State Diagrams

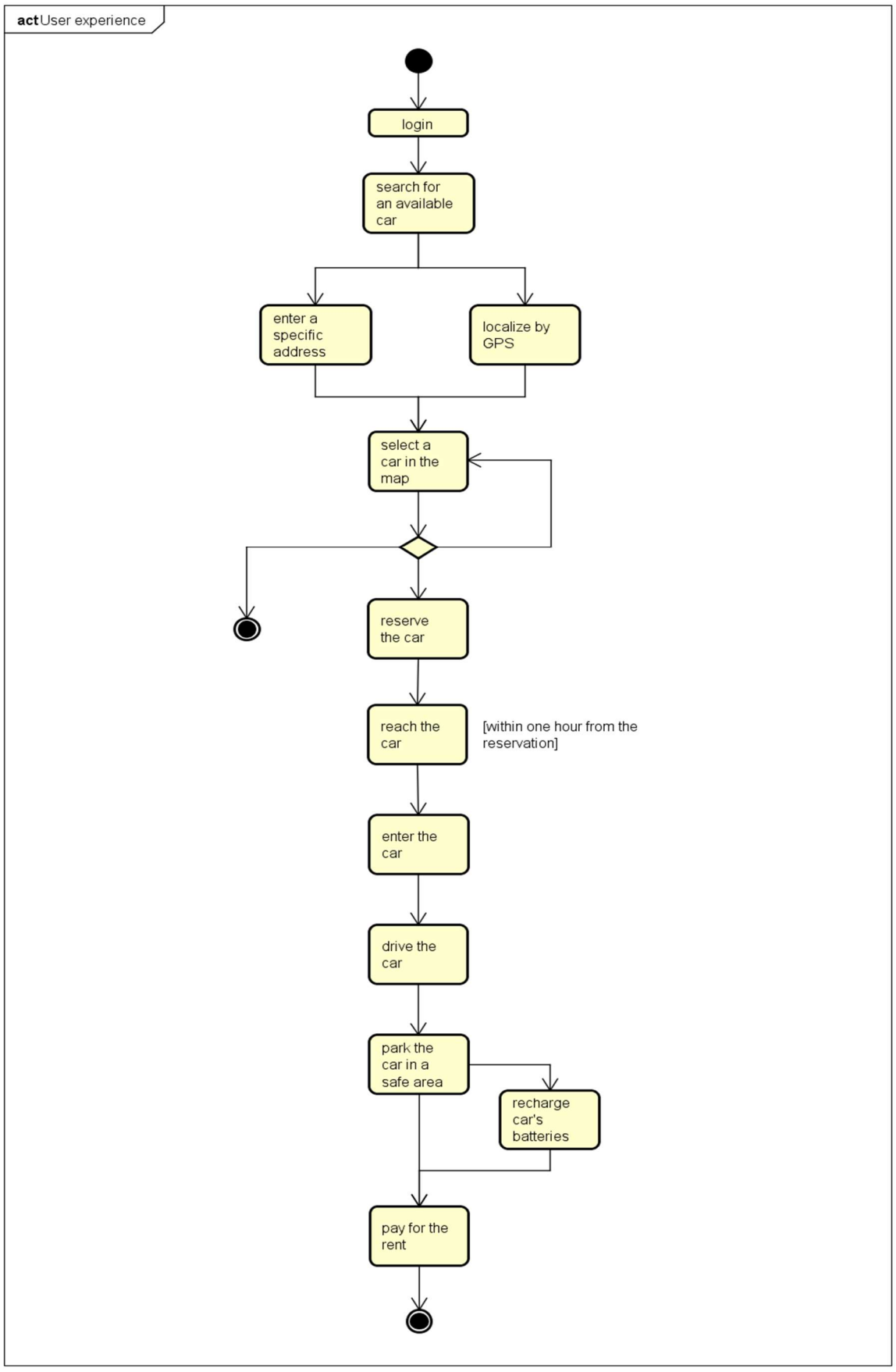
### 5.4.1 Car




### 5.4.2 User



## 5.5 Activity Diagram



# 6. ALLOY

```
module powerEnjoy

/**---------------------------------ID-----------------------------------**/

abstract sig ID {}{ID in UserID + CarID +ZoneID}

--SetDefinition: Each ID extension is a specified value of a certain signature
sig UserID, CarID, ZoneID extends ID{
}{UserID = User.userID and CarID=Car.carID and ZoneID=Zone.zoneID}

/**-----------------------------BOOLEAN-------------------------------**/
--SetDefinition: The booleans are used only in ParkedCar, FreeCar and ChargingPosition
abstract sig Boolean{
}{Boolean in ChargingPoint.free+CarStatus.chargingBattery}

sig True extends Boolean{
}{#True<=1}

sig False extends Boolean{
}{#False<=1}

/**-----------------------------POSITION-------------------------------**/

--SetDefinition: Each Position is contained in the union of portionMap and contained position
sig Position{
}{Position in Zone.portionMap+Zone.containedPosition}

/**---------------------------------TIME-----------------------------------**/

--SetDefinition: Each time belogns to a Employment
--ValDefinition: There are not negative times
sig Time{
   timeLog: one Int
}{timeLog>=0 and
      Time in Employment.reservationTime+Employment.startUse+Employment.endUse}

/**--------------------------------SYSTEM---------------------------------**/

--ValDefinition: There is only one system
sig System{
   map: one Map,
   userTable: set User,
   carTable: set Car,
   employmentSet: set Employment
}{#System = 1}
```

```
/**----------------------------------CAR-----------------------------------**/

--SetDefinition: Each car is contained in the System car Table
--                Each car Position belongs to the contained position set of a zone
--ValDefinition: The battery value goes from 0 to 100 in order to identify a percentual
sig Car{
   carID: one CarID,
   carPosition: one Position,
   currentBattery: one Int,
   carStatus: one CarStatus
}{Car in System.carTable
      and carPosition in Zone.containedPosition
        and currentBattery>=0 and currentBattery<=100}

--Fact Definition: Each car is unique and this unicity is specified by a different ID
fact carUnique{
   all disj c1, c2: Car| no c1.carID & c2.carID
}

--Fact Definition: There is a univocal correspondation between a Car and a Status,
--                  each Car has a different Status
fact correspondingStatusCar{
   all disj c1,c2: Car | no c1.carStatus & c2.carStatus
}

--Fact Definition: There aren't two Car in the same Position
fact noCarSamePosition{
   no disj c1,c2 : Car | c1.carPosition = c2.carPosition
}

--FactDefinition: If a Car is Free it couldn't have an employment
fact noFreeCarImployed{
   all c:Car| c.carStatus in FreeCar iff
      no e:Employment | e.carEm= c
}

--FactDefintion: A reserved Car implies an employment with no startUse and endUse
fact ReservedCarImployed{
   all c:Car| c.carStatus in ReservedCar iff
      one e:Employment | e.carEm= c and #e.startUse=0 and #e.endUse=0
}

--FactDefintion: An occupied Car implies an employment with a startUse but no endUse
fact OccupiedCarImployed{
   all c:Car| c.carStatus in OccupiedCar iff
      one e:Employment | e.carEm= c and #e.startUse=1 and #e.endUse=0
}

--FactDefintion: A parked Car implies an employment with a startUse and endUse
fact ParkedCarImployed{
   all c:Car| c.carStatus in ParkedCar iff
      one e:Employment | e.carEm= c and #e.startUse=1 and #e.endUse=1
}
```

```
--FactDefinition: A ParkedCar can be charged only in ChargingPoint
fact ParkedAndCharging{
   all c:Car | all ch: ChargingPoint | c.carStatus in ParkedCar+FreeCar+ReservedCar
      and c.carPosition in ch.chargingPosition
         implies (c.carStatus.chargingBattery = True or c.carStatus.chargingBattery = False)
            else c.carStatus.chargingBattery = False
}

/**-----------------------------CARSTATUS-------------------------------**/

--SetDefinition: Each CarStatus belongs to a Car
abstract sig CarStatus {
   chargingBattery: one Boolean
}{CarStatus = Car.carStatus}

sig FreeCar extends CarStatus{
}

sig ReservedCar extends CarStatus{
}{}

sig OccupiedCar extends CarStatus{
}{chargingBattery = False}

sig ParkedCar extends CarStatus{
}

/**-------------------------------USER---------------------------------**/


--SetDefinition: Each User belongs to the System User Table
--                Each User Position belongs to the contained position set of a zone
sig User {
   userID : one UserID,
   userPosition: one Position,
   userStatus: one UserStatus
}{User in System.userTable
      and userPosition in Zone.containedPosition}

--FactDefinition: Each User is Unique
fact userAreUnique{
   all disj u1, u2: User| no u1.userID&u2.userID
}

--FactDefinition: Each User has a different Status
fact correspondingStatusCar{
   all disj u1,u2: User | no u1.userStatus & u2.userStatus
}

--FactDefinition: If a registred User couldn't have an employment
fact noRegistredUserImployed{
   all u:User| u.userStatus in RegistredUser iff
      no e:Employment | e.userEm= u
}
```

```
--FactDefinition: A user is reserving a car only if there is a employment with his name
fact ReservingUserImployed{
   all u:User| u.userStatus in ReservingUser iff
      one e:Employment | e.userEm=u  and #e.startUse=0 and #e.endUse=0
}

--FactDefinition: A user is reserving a car only if there is a employment with his name
--                and he'using the car
fact OccupiedUserImployed{
   all u:User| u.userStatus in OccupiedUser iff
      one e:Employment | e.userEm=u  and #e.startUse=1 and #e.endUse=0
}

--FactDefinition: A user is reserving a car only if there is a employment with his name
--                and he's completed the employment
fact PayingUserImployed{
   all u:User| u.userStatus in PayingUser iff
      one e:Employment | e.userEm=u and #e.startUse=1 and #e.endUse=1
}

/**----------------------------USERSTATUS-------------------------------**/

--SetDefinition: Each UserStatus belongs to a User
abstract sig UserStatus {
}{UserStatus= User.userStatus}

sig RegistredUser extends UserStatus{
}

sig ReservingUser extends UserStatus{
}

sig OccupiedUser extends UserStatus{
}

sig PayingUser extends UserStatus{
}
```

```
/**---------------------------------MAP---------------------------------**/

--SetDefinition: A map belongs to a System
sig Map{
   zoneSet: some SafeZone,
   borderSet: set BorderZone
}{Map in System.map}

/**---------------------------------ZONE---------------------------------**/

--SetDefinition: Each zone is contained in zoneSet or borderSet
--FactDefinition: Each zone is delimited by at least three position (triangle)
abstract sig Zone{
   zoneID: one ZoneID,
   portionMap: some Position,
   containedPosition: set Position
}{ Zone in Map.zoneSet+Map.borderSet and #portionMap>=3}

--SigDefinition: A SafeZone could have a charging point
sig SafeZone extends Zone{
   powerStation: lone PowerGridStation,
}

--SigDefinition: A borderZone is a zone where a car can go but where it can't be parked
sig BorderZone extends Zone{}



--FactDefinition: There aren't intersection of position between two Zones
--                 Each zone is unique, so they have a different ID
fact noDataIntersection{
   all disj z1,z2: Zone|  no z1.portionMap & z2.portionMap
      and no z1.containedPosition & z2.containedPosition
            and no z1.containedPosition & z2.portionMap
               and  no z1.zoneID & z2.zoneID
}
--FactDefintion: There is no intersection between containedPositionSet and portionMap set
fact portionMapContainedPositionDisjointed{
   all z: Zone| no z.portionMap & z.containedPosition
}


/**----------------------POWERGRIDSTATION---------------------------**/
--SetDefinition: Each PowerGridStation is contained in a SafeZone
sig PowerGridStation{
   chargingGrid: some ChargingPoint
}{PowerGridStation in SafeZone.powerStation}
```

```
/**-------------------------CHARGINGPOINT-----------------------------**/

--SetDefinition: Each ChargingPoint is contained in a PowerGridStation
sig ChargingPoint{
   free: one Boolean,
   chargingPosition: one Position
}{ChargingPoint in PowerGridStation.chargingGrid and
      chargingPosition in SafeZone.containedPosition}

--FactDefinition: Every ChargingPoint is contained in the SafeZone that has its
--                own PowerGridStation
fact stationPointPositionCorrespondation{
    all z: SafeZone| all c: ChargingPoint|
      c.chargingPosition in  z.containedPosition
         iff c in z.powerStation.chargingGrid
}

--FactDefinition: If there is a Car in the same position of the ChargingPoint
--                we assume this chargingPoint occupied
fact notFreeMeansOccupiedByACar{
   all ca:Car | all ch: ChargingPoint|
       ca.carPosition = ch.chargingPosition  iff ch.free=False and
      !(ca.carPosition = ch.chargingPosition)  iff ch.free=True
}

--FactDefinitoin: Each Charging Point has a unique position
fact noChargingPositionShared{
   all disj c1,c2: ChargingPoint| no c1.chargingPosition & c2.chargingPosition
}

/**---------------------------EMPLOYMENT-------------------------------**/

--SetDefinition: Each Employment belongs to the Employment Set
--ValDefinition: reservation Time is anterior of start Use Time
--               startUseTime is anterior of end Use Time
sig Employment{
   userEm: one User,
   carEm: one Car,
   reservationTime: one Time,
   startUse: lone Time,
   endUse: lone Time
}{Employment in System.employmentSet
      reservationTime.timeLog <= startUse.timeLog
        and startUse.timeLog <=endUse.timeLog}

--FactDefiniton: Each employment is unique and there aren't two employments
--                that involve the same user or car
fact employmentUnicity{
   no disj e1,e2 : Employment | e1.userEm = e2.userEm or
      e1.carEm = e2.carEm
}
```

```
/**----------------------------ASSERTIONS-------------------------------**/

--AssertDefintion: if a Car has a FreeStatus then it cannot have an employment
assert oneCarOneEmpoyment{
   all c:Car| !(c.carStatus in FreeCar) implies
      (one e:Employment| e.carEm = c)
}

--AssertDefintion: if a User has a RegistredUser then it cannot have an employment
assert oneUserOneEmployment{
   all u:User| !(u.userStatus in RegistredUser) implies
      (one e:Employment| e.userEm = u)
}

--AssertDefinition: Each reservation corresponds to a specified carStatus and
--                  a userStatus for example if a reservation is completed (it has
--                  a endUse Time) the car is parked and the user is paying
assert relationBetweenCarUserAndEmployment{
   all e:Employment |
      (#e.reservationTime=1 implies
         e.carEm.carStatus in ReservedCar+OccupiedCar+ParkedCar and
         e.userEm.userStatus  in ReservingUser+OccupiedUser+PayingUser) and
      (#e.reservationTime=1 and #e.startUse=1 implies
         e.carEm.carStatus in OccupiedCar+ParkedCar and
         e.userEm.userStatus  in OccupiedUser+PayingUser) and
      (#e.reservationTime=1 and #e.startUse=1 and #e.endUse=1 implies
         e.carEm.carStatus in ParkedCar and
         e.userEm.userStatus in PayingUser)
}

--AssertDefinition: This assertion verifies the correct number of signatures of each
--                  possible representation
assert correctQuantities{
   #CarStatus=#Car
   #UserStatus=#User
   #Position >= #Zone+ #Zone+ #Zone+ #containedPosition
   #Boolean <= 2
   #Employment <= #Car
   #Employment <= #User
   #Time >= #Employment
}
```

## 6.1 Alloy Execution

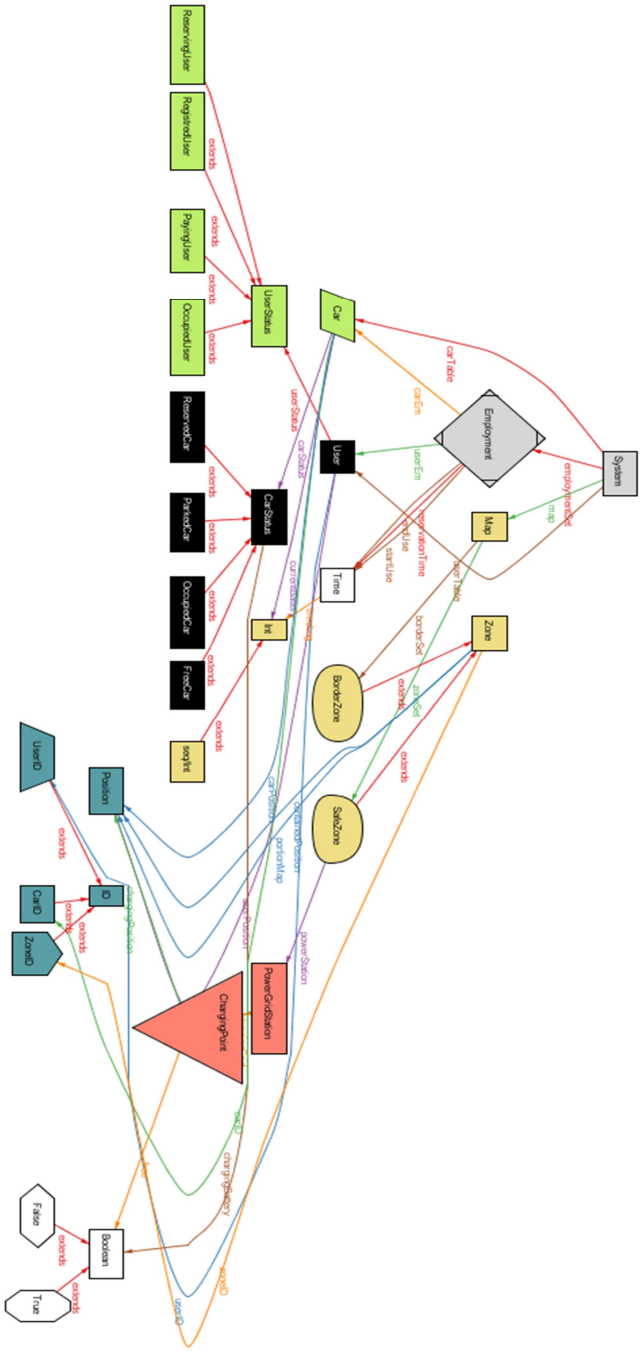
```
check oneCarOneEmpoyment
check oneUserOneEmployment
check relationBetweenCarUserAndEmployment
check correctQuantities


pred show[]{
#User>0
#BorderZone>0
#SafeZone>0
#Car>0
#Employment>0
#PowerGridStation>0
}

run show for 8
```

## 6.2 Alloy Results

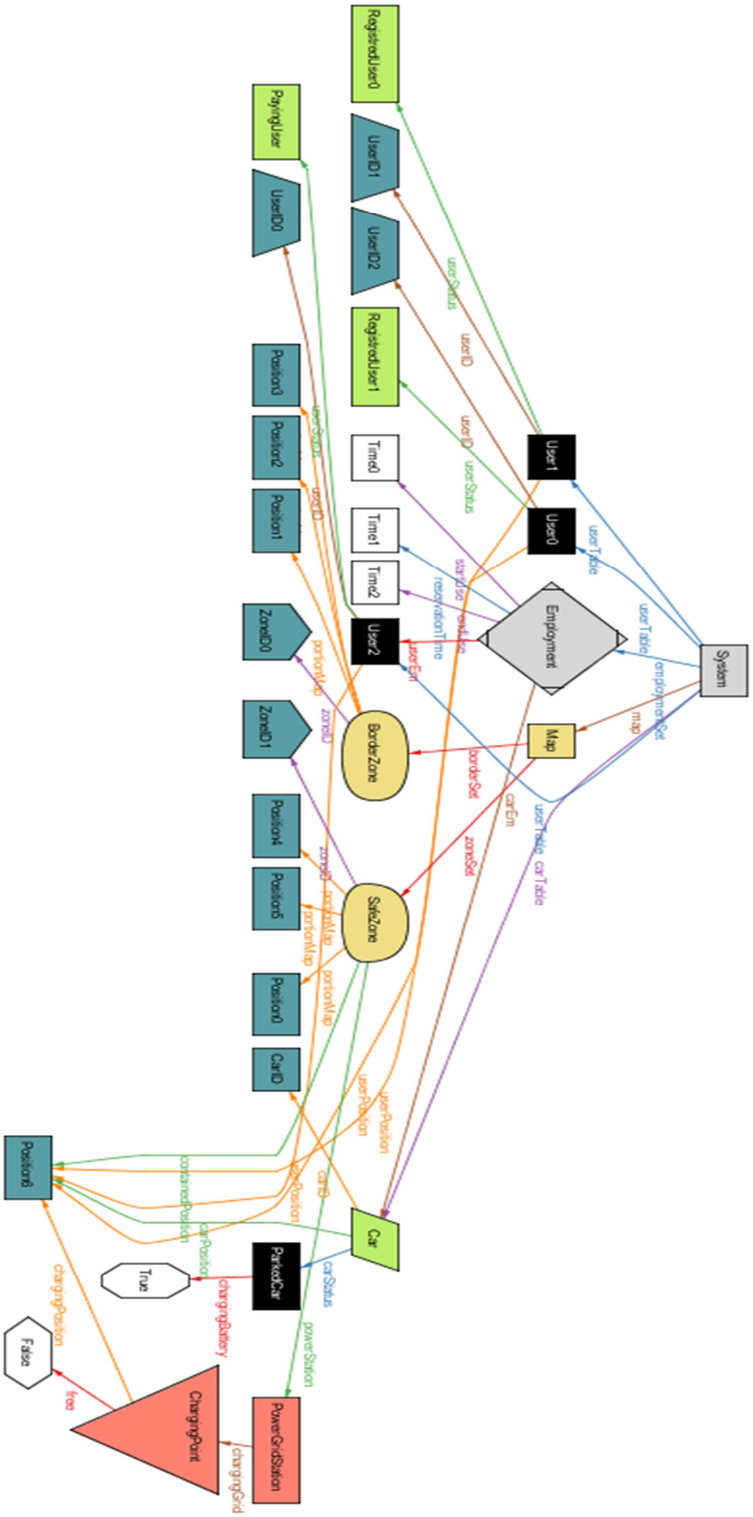
```
5 commands were executed. The results are:
  #1: No counterexample found. oneCarOneEmpoyment may be valid.
  #2: No counterexample found. oneUserOneEmployment may be valid.
  #3: No counterexample found. relationBetweenCarUserAndEmployment may be valid.
  #4: No counterexample found. correctQuantities may be valid.
  #5: Instance found. show is consistent.
```

## 6.3 Alloy Diagrams

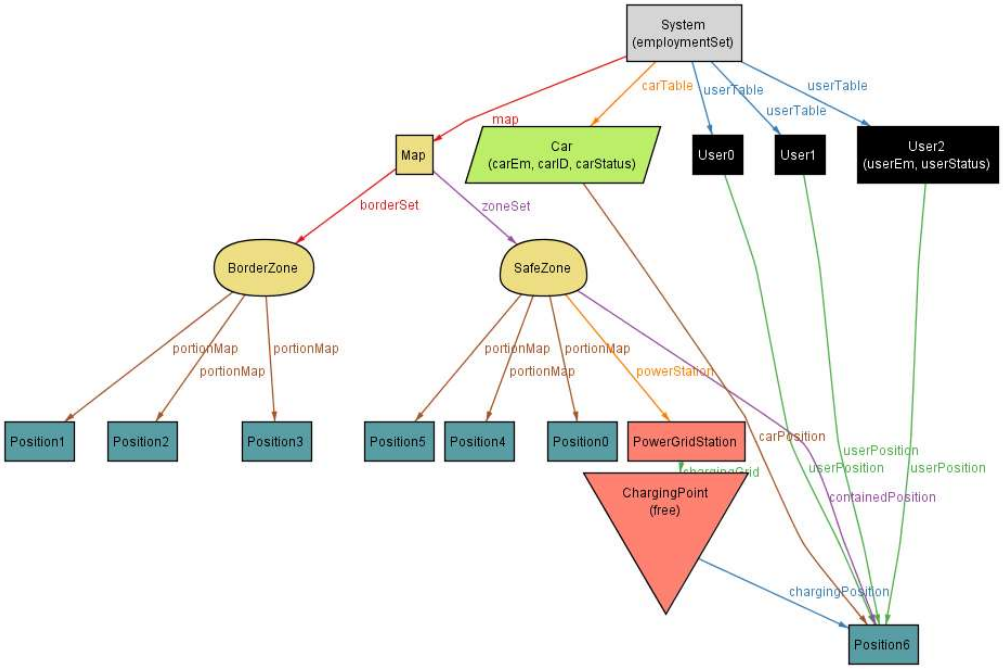
### 6.3.1 MetaModel



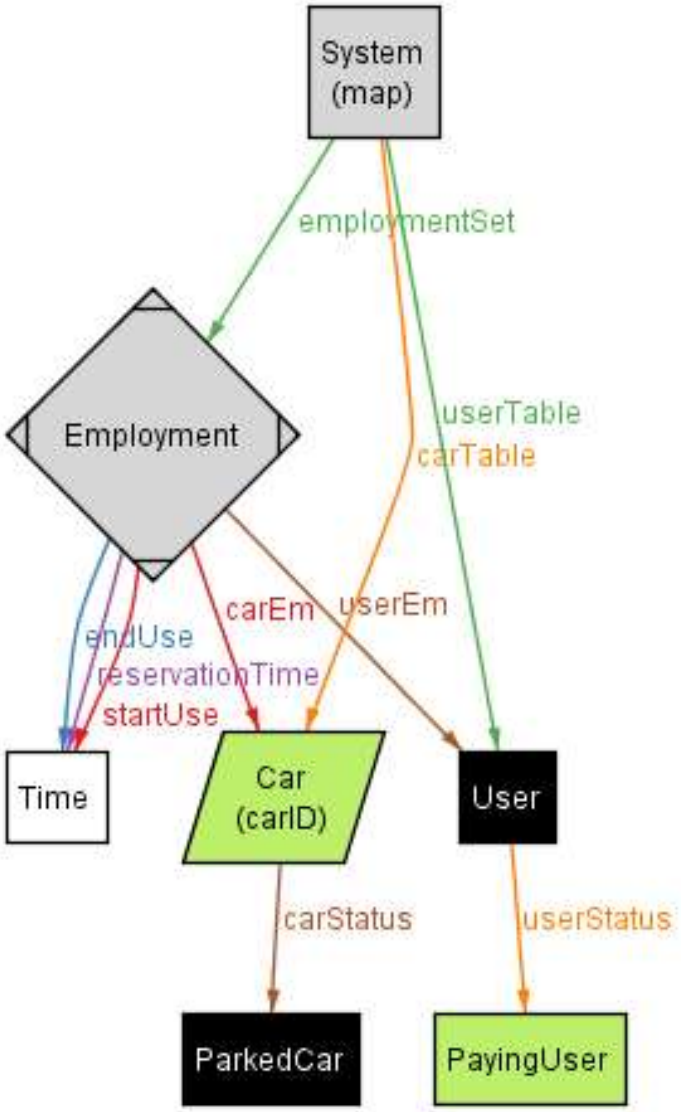
6.3.2 World


RegistredUser0
PayingUser
UserID1
UserID0
UserID2
RegistredUser1
Position3
Position2
Position1
Time0
Time1
Time2
User1
User0
User2
Employment
System
ZoneID0
ZoneID1
BorderZone
Map
Position4
Position5
SafeZone
Position0
CarID
Position6
Car
True
False
ParkedCar
ChargingPoint
PowerGridStation
userStatus
userID
userTable
employmentSet
map
borderSet
zoneSet
carEm
carTable
userPosition
carID
carPosition
carStatus
zoneStation
chargingStation
chargingPosition
free
chargingGrid

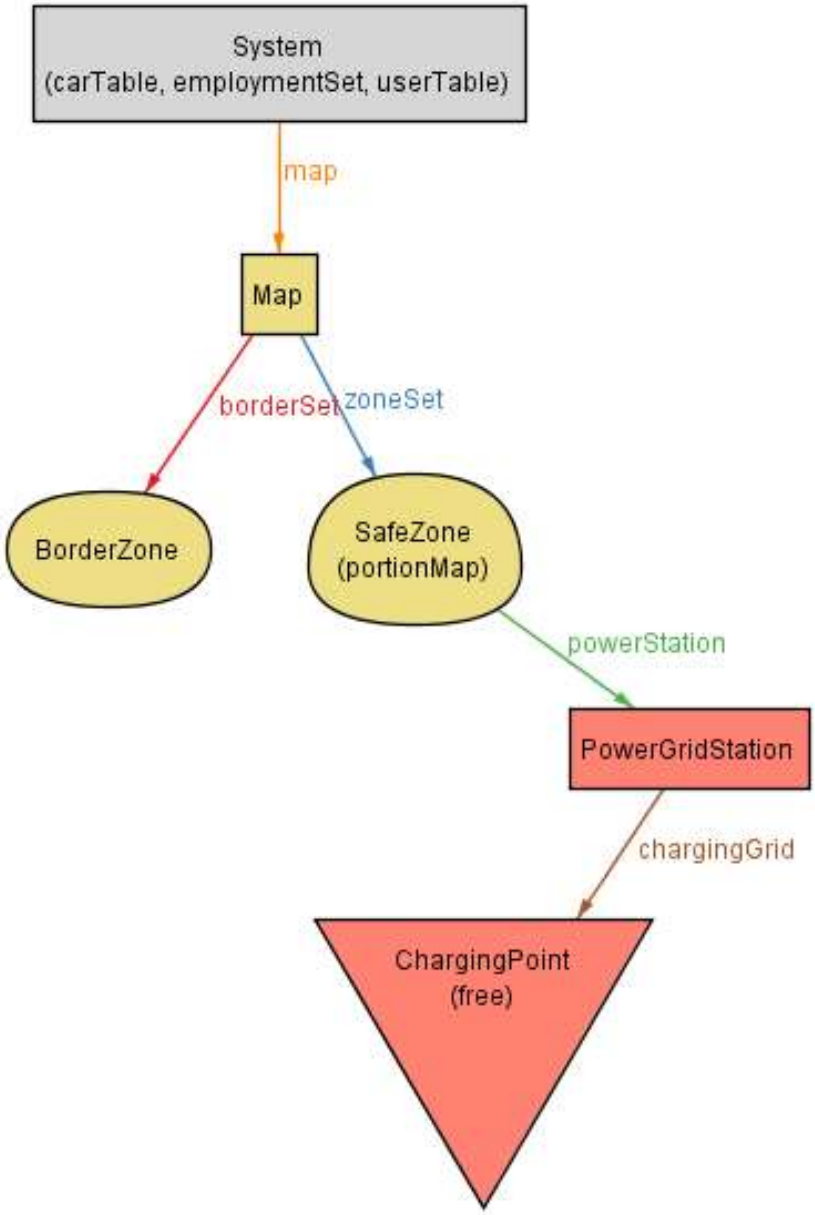
### 6.3.3 Partial Representation




This image represent a simplified version of the world generated, it's projected over the employment and the status between the car and the User2.




This image represent an employment characterized by a user that has parked the car and he's going to pay the rent of the car, and it' projected over the position of the user and the car.




This image represents the composition of a map without the representation of the contained positions and the positions that delimit the zones (Zone.portionMap).

# 7. APPENDIX

## 7.1 Software and tool used

- Alloy Analyzer 4.2
- Gimp 2.8
- Signavio Academics for use cases
- Lucidchart for class diagram
- Astah Professional for sequence diagrams, state chart diagrams and activity diagrams

## 7.2 Hours of works

Giacomo Bossi:

- 17/10: 1hr
- 25/10: 2hr
- 27/10: 2hr
- 29/10: 3hr
- 31/10: 3,5 hr
- 2/11: 4 hr
- 3/11: 5 hr
- 4/11: 4 hr
- 7/11: 3,5 hr
- 8/11: 4 hr
- 9/11: 3 hr
- 10/11: 3hr
- 11/11: 3 hr

Marco Nanni:

- 17/10: 1hr
- 25/10: 2hrs
- 27/10: 2hrs
- 31/10: 4hrs
- 03/11: 4hrs
- 04/11: 4hrs
- 05/11: 5hrs
- 06/11: 4hrs
- 07/11: 6hrs
- 08/11: 5hrs
- 09/11: 3hrs
- 10/11: 2hrs
- 11/11: 2hrs