Politecnico di Milano

Software Engineering 2

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT

PowerEnjoy

Authors:

Simone Bruzzechesse

Luca Franceschetti

Gian Giacomo Gatti

INDEX

1 INTRODUCTION

1.1 PURPOSE

This document represents the Requirements Analysis and Specification Document (RASD). Its aim is to capture all the functional and non-functional requirements that the system-to-be must respect, to satisfy the stakeholders goals, under certain domain properties. This document also contains Use Case Diagrams, Sequence Diagrams and Class Diagrams that can be useful to better understand how the system is organized. Further, this document is a valid basis for system testing, verification and validation and has also a contractual value.

1.2 SCOPE

The aim of our project is to develop a digital management system for a car-sharing service that exclusively employs electric cars. The system should provide the functionality normally provided by car-sharing services. Users must be able to register to the system by providing their credentials and payment information, then they receive back a password that can be used to access the system. Registered users must be able to find the locations of available cars within a certain distance from their current location or from a specified address. The system provides also the possibility to reserve a single car, but with some constraint: for example, if a car is not picked up, the user must pay a fee. On the other hands, if a user reaches a reserved car, he must be able to tell the system he's nearby his reserved car, so the car will be unlocked and the user can enter and start his rent.

Car-sharing system initiate the charging of money as soon as the engine ignites, and the system starts charging the user for a given amount of money per minute. Indeed, the user is notified of the current charges through a screen on the car. The system stops charging the user as soon as the car is parked in a safe area and the user exits the car.

The set of safe areas for parking cars is predefined by the management system, so we can contact a database to catch some information about the current position of the car, and then the system can decide if it is parked in a safe area.

Although, the system must be able to define certain user's behaviour with the car-sharing services and apply some discount (or charging) in consequence of determinate action.

Users will be able to use a mobile application for use the car-sharing services and register himself during the first rent, and to register himself with a web application created to improve the comfort of registration.

1.3 GOALS

1. Let the users create a profile.
2. Let the users logging in into the system and managing their profile.
3. Let the users research a car.
4. Let the users reserve a car.
5. Let the users decline reservation for a car.
6. Let the users unlock and use a car.
7. System should inform users about car and trip status during the ride.
8. System should encourage environmental and system friendly behaviour of users applying discounts on rides.
9. System discourages user's bad behaviour regarding the system management by applying charges to the user's trip.

1.4 ACTORS

Actors of our system are essentially two, even if the second one is much more important and assume different states according to how he/she is interacting with the system.

- **Guest**: a guest is a person who is not registered in the system yet. He cannot use features of the system until he/she sign up.
- **User**: a user is a person that has already signed up into the system. When a user logs in, we mean him/her as a LOGGED USER. The logged user can take advantage of every feature of our system, and depending on what his actions are he/she can be a simple user or a driver.

1.5 DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.5.1 Definition

We'd like to specify the meaning of some words that we usually use in this document to avoid misunderstanding and confusion.

- USER: a user is a person that has already signed up into the system.
- GUEST: a person who is not registered in the system.
- PASSENGER: a person, not necessarily a user, who shares a car with other guests or users (at least one user is requested to drive the car).
- RESERVATION: we mean when a user wants to reserve an available car, sending a request to the system through the mobile application.
- CAR: is considered an electric car, registered into the system. It's connected with mobile application thanks to different technologies. It's always kept under control by our system. Maintenance costs are not on users.
- DISCOUNT: system applies discount to encourage users' good behaviour. These are percentage discount applied at the end of the ride on the full price. In case of multiple discounts, then they are applied starting from the one with lower percentage.

- CHARGES: system applies charges to the user's trip in order to avoid bad behaviour of this one regarding the management of the system.
- TRIP: it indicates the travel of the user with an electric car, it presupposes that the user has already reserved the car to use it.
- PLUG: it is a power point where the user can plug the car to recharge its battery.
- CHARGING STATION: a place where there are a certain number of plugs and a certain number of car that are plugged.
- PLUG-IN THE CAR: since system uses only electric cars, they must be plugged in to be charged. System provides different plug-in station around the city for the users: they are reported on the map and can be used only by electric car in the system.
- CHARGING STATION AREA: a safe area within 3km of radius from a certain charging station, it is useful to identify some discount in the system.
- SAFE AREA: cars must be left in a safe area. By safe area we mean approximately a circle area within a range of 15 km from the city centre.

1.5.2 Acronyms

We'd like to explain some acronyms we use in this document to make reading easier.

- RASD: Requirements Analysis and Specification Document.
- GPS: Global Positioning System.
- DBMS: Database Management System.
- SD: Sequence Diagram.

1.6 STAKEHOLDERS

If the PowerEnjoy project is intended as an academic exercise the stakeholder is the professor who gave us the project. The stakeholder expects to receive a complete documentation that fulfils the requests and the main functionality of the system to be.

On the other hand, if our system is intended as a “real system” to develop than our stakeholders are:

- **Users**: they are involved in our project because they will use our system, in particular they will have the possibility to choose between the web application or the mobile application.
- **Car-sharing service**: our system will offer a web application and a mobile application, developed to manage in a better way all the car-sharing system provided by related company.
- **System administrators**: they are the managers of our system.
- **Testers**: they should check if our system respects all the requirements that have been identified.

1.7 REFERENCE DOCUMENTS

- International standard IEEE Systems and software engineering: Life cycle processes — Requirements engineering (ISO/IEC/ IEEE 29148).
- AA 2016-2017 Software Engineering 2 — Project goal, schedule, and rules.

1.8 OVERVIEW

Our document is organized in four main parts:

- **Introduction**: in this section, we give an overview of the scope and goals of our system-to-be. We also identify the main actors that will be involved in our system and give the basic definitions of some words we will often use in this document.
- **Overall description**: in this part, we try to focus our attention on constraints and assumptions concerning our system-to-be and the world around it. This section also considers some possible future implementations that could be added to our system.
- **Specific requirements**: this section is the body of our document. All the specific requirements that our system need are described here and they are associated with different kinds of diagrams, to create a model of the real system.
- **Bibliography**: in this part, we specify the documents or books we have referred to.

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The product is a web based application that interacts with a back-end server that handles data and stores them into a DBMS. The client application is supposed to be developed in both web and mobile form, but they include some difference. There are some main features included:

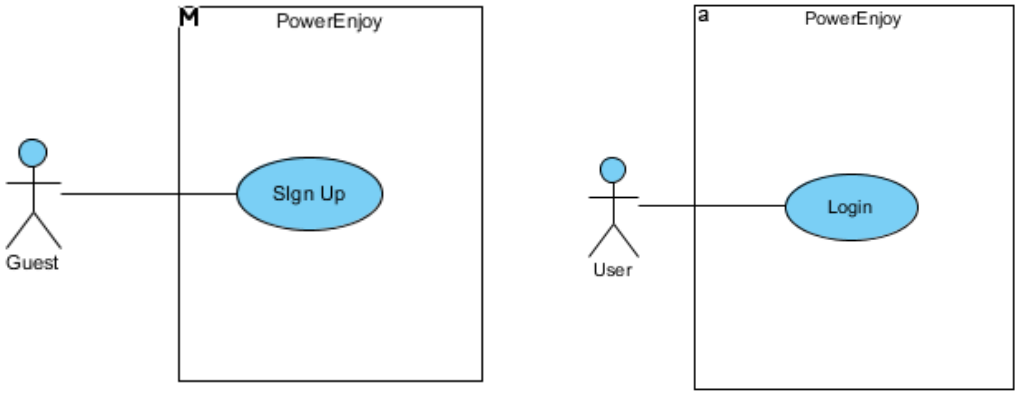
- Multi-platform: the back-end system can be used to develop the mobile and web applications.
- User account: the system allows the user to create an account in the system and provide features of updating and viewing profiles.
- Number of users being supported by the system: though the number is precisely not mentioned, the system can support many online users at a time.
- Car reservation: the system allows the reservation of a car, locking the car until the user reach the car, or otherwise until the reservation is cancelled.
- "Money saving" option: the system can help users with some useful advice, in order to obtaining discounts thanks to good behaviour. If the user enable this option, the system asks for a specific destination and calculate some option for leave the car in a "safe area" plugged into the electric grid.

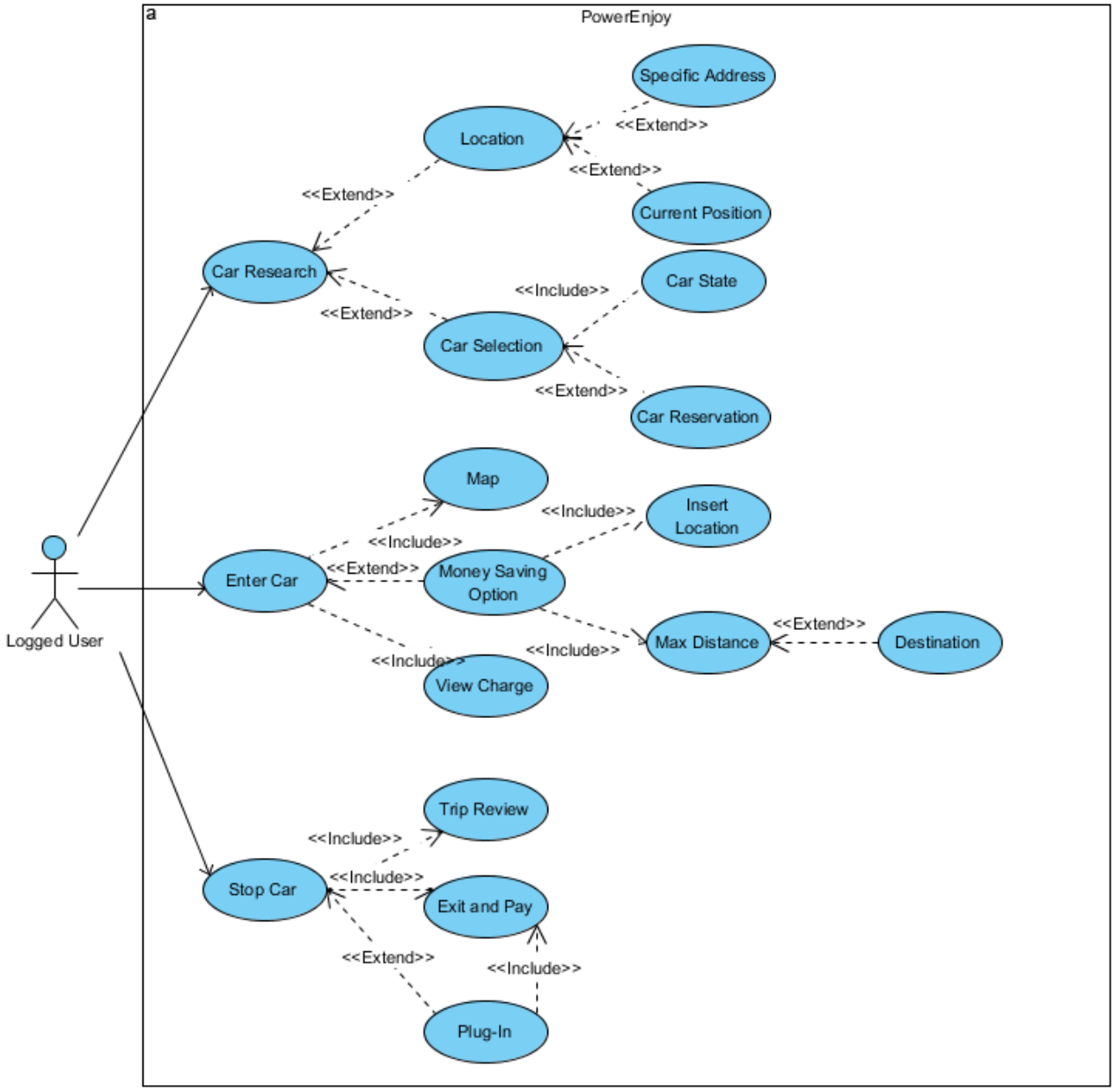
2.2 PRODUCT FUNCTIONS

The main function of the systems is to allow users to reserve an electric car, provided by our car-sharing service. The user can also manage his/her reservation, deleting or confirming it. System encourages users' environmental friendly behaviour providing them discount in case of trip with at least 2 passengers. It also provides special deals if the users take care of plugging-in the car on the electric grid because he/she contribute to the wellness of the system.

- Users:
    - Sign up into the system
    - Log into the system
    - Manage account
    - Research cars
    - Select car
    - Reserve car
    - Delete reservation
    - Unlock the car to use it
    - View car status
    - View charge during the trip
    - Enable “Money saving” option
    - Plug-in the car in order to get the discount
    - Visualize “trip review”
    - Conclude the rent and pay

2.2.1 Use Case Diagrams





2.3 USER CHARACTERISTICS

We expect that our user's application will be installed by users who are looking for an easy way to research or reserve a car, with the formula of car-sharing services. The system provides the possibility to reserve a car, check their reservations whenever they want, spot some car roundabout through user's position, check the discount with "money saving" option and make more comfortable the payment of the service. The

system is developed to make the user comfortable with the use of the mobile application, except for the registration, where the user can use also web application, that is more conformable because are requested some personal data, that could be difficult to insert with the keyboard of mobile phones.

2.4 CONSTRAINTS

2.4.1 Regulations

The system must require permission to use GPS position of user's mobile phone. Also, must manage personal sensible data (like personal data on the registration's database or mobile phone number used for communicated with users) respecting current privacy law. System's communications or notification must not frustrate the user, sending it only when it's necessary (it must be send an appropriate number of email, without SPAM) and it must contain useful information about the service.

2.4.2 Reliability requirements

The system should stay online and available always, excepted for maintenance break (updating the system, for example). Otherwise the user won't be able to perform reservation's requirement or rent, and car-sharing service lose money and client.

2.4.3 Hardware limitation

- Web app
    - User must use a device with active internet connection, and with a browser that support modern HTML pages, in order to display the registration form in an appropriate way.
- Mobile application: to use the service with mobile phone application, user should use a smartphone with some specs:
    - 3G connection or WIFI connection
    - GPS
    - Free space on smartphone's memory for save application's data

2.4.4 Criticality

All the car of the service must be connected to the operative centre with internet connection, because the only way to localize one of them is send precise location with connection. If a car cannot communicate its info to the system, we could considerate it as unavailable. No user will be able to use it, unlock it or reserve it.

2.4.5 Performance requirements

- The system must display available cars in a certain area in a reasonable time, to allow user to decide what car pick up without losing time.
- The system must be able to query a big number of car, picking battery's data and position, and display the whole information on mobile application almost in real time.
- System should calculate rapidly eventual discount because the payment is committed exactly at the end of the rent, so the sums displayed on car's screen must be coherent with the sum in the payment.
- The request about reserving a car by one user must be scheduled, because the system must manage the concurrency between users. After the scheduling, the system should calculate as soon as possible the user that have truly reserved the car, and communicate respectively that the booking was successful or not.

2.5 Assumptions and Dependencies

We want to clarify some ambiguous specification given in the assignment document and we assume as true some situation in order to simplify the model of the software.

- We deliver the payment management at the end of the trip to an external system that is able to handle this situation, so it can manage also some exceptions that can occur during the payment. For instances the user could not have enough money to conclude the transaction.
- We assume that the fee is payed as soon as the reservation expires.

- The payment of the trip is made as soon as the user leaves the car if he doesn't select the possibility to plug the car (or if he is not able to do that), while if he chooses this option the payment is postponed of 2 minutes because the system allows him to plug the car. Then if the system, after two minutes, detects that the car is plugged it will apply the discount, otherwise not.
- Since "the user leaves the car" and "the user is near the car" are ambiguous, we assume that a user is near the car or is leaving it if the distance (calculated from the two GPS) between them is smaller/bigger than a certain amount.
- We assume that the charging stations are always available and perfectly working. If not, we assume that they are immediately repaired by the employees of the system.
- Since we exactly know, thanks to Placemeter, how many people are actually in the car we can apply the 10% discount only to part of trip where the passengers were at least 2. For instances if during the first half of the trip there were 3 passengers on the car and then they were dropped off, so the user finishes the trip on his own, then the discount is applied only to the first half of the trip.
- We assume that if one car's charge is less than 5%, then it encourages the user to stop the trip as soon as possible. As the car is left it won't be able for others trip until one of the employee of the system take care of bringing it to the nearest charging station and plugging it. So, when the car's charge will be again upper than a certain percentage it will be available again for the users.

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 User Interfaces

The user interface models are discussed later one during the presentation of the use cases, in particular every use case is associated to the corresponding page of the user interface.

3.1.2 Hardware Interfaces

Placemeter

We'd want to adopt this system to count the number of people in the car. The knowledge of this amount it's important because system applies some discount even according to the number of passenger in the car. No one wants cameras staring at us, yet the truth is that making those counts can yield useful insights for society. At least that's what Placemeter, a young company founded by two French entrepreneurs, thinks. That's why it built robust computer vision technology to make the counts. Today, Placemeter is revealing its new people counting sensor. As we said, it's a sensor: it's a lens attached to a computer that runs the analysis "at the edge of the network," and the only data that comes out the other end are counts of what the computer saw. The output looks like a spreadsheet. In terms of looks, the device doesn't have the appearance of a camera, either. It is meant to attach to the edge of a window. It looks out through a one-way mirror, which also makes the device more discreet to those outside. The sensor can send data to Placemeter's servers via wi-fi or GSM. Either way, its data load is very light.

(http://observer.com/2015/06/first-look-new-sensor-can-count-people-and-cars-without-video/)

3.1.3 Software Interfaces

Both part of the system, mobile application and web application, use the same server resource to fetch data and save it. So server side must be compatible with different platform. We propose a possible implementation of the system:

- DBMS:
  - MySQL
- Server side:
  - JBoss application server
- Operating system
  - Android OS for mobile phone
  - Windows 10 for personal computer
- Browser
  - Google Chrome (for guarantee the compatibility with HTML5)

3.1.4 Communication Interfaces

The client application communicates with the back-end server via HTTPS (port 443). The back-end services connect to the DBMS on the default port.

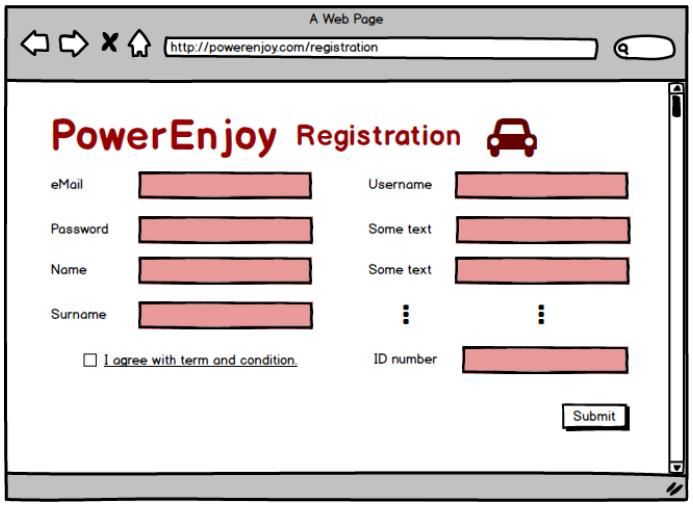
3.2 FUNCTIONAL REQUIREMENTS

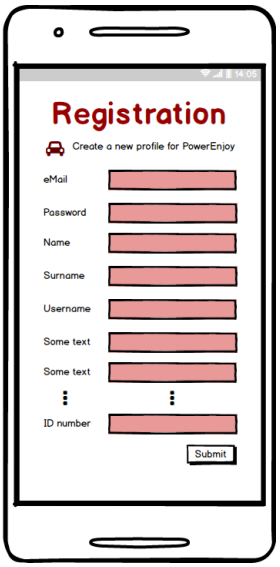
3.2.1 Sign Up

Guests can subscribe to the system via mobile or web application. In both cases, they should fill a form with their personal data and authorize the recipient to use and process their personal details. If the guest accepts the conditions then he can complete the registration, otherwise it is cancelled. Once a guest send the registration, the system checks the consistency of the data inserted by the guest and if everything is correct a mail and a message containing two codes are sent to the new user. Then the guest must insert in a new page the two codes he received and perform a submission, if the data he has submitted are right then the system complete the registration of the user.

Actor	**Guest**
Goal	Goal n°1
Input Condition	The guest isn't registered to the website
Event Flow	1. The guest enters the website; 1.1. The guest enters the application; 2. The guest clicks on the "SIGN UP" button; 3. The guest fills in the form where he must write: 3.1. Name 3.2. Surname 3.3. Email 3.4. Username 3.5. Password 3.6. Date of birth; 3.7. Telephone number 3.8. Address 3.9. Picture 3.10. **Driving license data**: 3.10.1. date of achievement 3.10.2. expire date 3.10.3. number 3.10.4. category 3.11. **Credit card data** 3.11.1. number 3.11.2. expire date 3.11.3. CVV 4. The guest accepts the conditions of the system; 5. The guest clicks the "DONE" button; 6. The system send an email to the user and a message on the mobile phone with two code to check the validity; 7. User insert in the application the codes and submit the request; 8. The system shows the home page of the user logged in;
Exit Condition	Registration is successfully done.

Exception	An exception can be caused if the username the guest inserts already exists or if some field that are not optional aren't filled. Another possibility of failure happens if the user doesn't accept the conditions provided by the system. There are also exceptions If the user doesn't insert the rights codes that were sent to the user and in case the information related to the credit card and driving license are wrong.



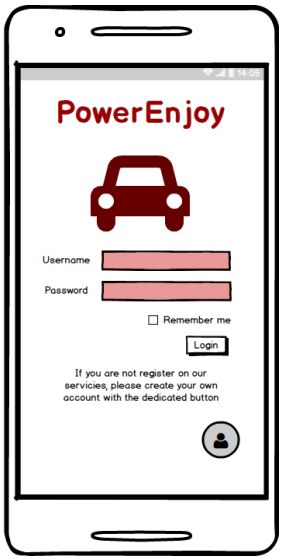


3.2.2 Login

Users must be logged in the system to use the functionalities of the application. During the login process users, must insert the username and the password. If the data they insert are right, then the users are redirected to their account page. In the 'user login' page, guests can be redirected to the 'registration' page.

Actor	**User**
Goal	Goal n°2
Entry Conditions	User has successfully signed up to the system

Event Flow	1. The user enters the application. 2. The user fills in the text fields in the home page with username and password. 3. The user tick the option in order to be logged even if he exit from the application. 3.1. The user doesn't tick the above option because he is not interest in this feature 4. The user clicks on the "LOG IN" button.
Exit Condition	The system shows the user his personal page.
Exception	The password and/or username inserted by the user are wrong. The System shows an error message to the user.



3.2.3 User Account Management

After the login process, users are redirected to their main page. Here they have the possibility to choose different actions to perform. For example, they can reset their password, change personal information, such the credit card associated to their account or update the driver license. Users can also delete their account.

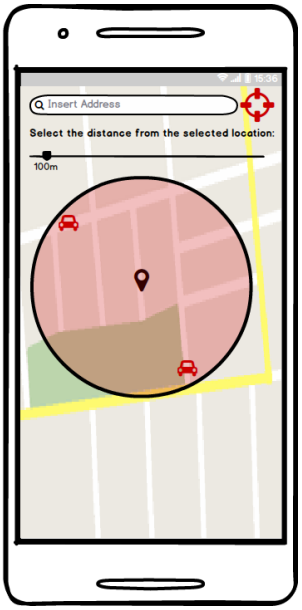
Actors	Logged User
Goal	Goal n°2
Input Condition	The user is logged into the system
Event Flow	1. The user select the personal a 2. The system shows at the users the modification form 3. The user modifies the chosen fields 4. The user submits the modifications 5. The system update the user's info
Output Condition	The user's profile is modified
Exception	The user doesn't submit the modifications; the new modifications are invalid. In both cases the new modifications are not applied

3.2.4 Research Car

When a user runs the application, he should be able to navigate the map in order to find the best alternatives for his/her choice. He should be able to set his current position with GPS or insert one manually. Map should allow user to zoom in and out to make the research easier and faster.

Actor	User
Goal	Goal n°3
Input condition	User is logged in the mobile application
Event flow	1. When the application is started, a map immediately appears on the screen with car that are displayed in the map as icon. Icons of available cars are green coloured, while icons of reserved but not picked-up cars are red. Picked-up cars are not shown in the map;

	2. User can navigate the map looking for the nearest available car.
Output condition	User selects the car he/she would like to reserve, so he/she is redirected to reservation page.
Exception	There are no available cars in the area.

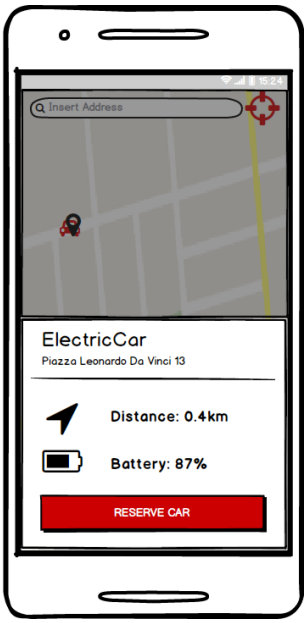


3.2.5 Select car, View Car Status and Reserve Car

User in the main page can search for the cars and then he can select one of them. Then the user can view the status of that car through the mobile application: specifically, he can check the status of the battery or if the car is reserved or not. Then he can also reserve the car.

Actor	User
Goal	Goal n°4
Input condition	User logged and find an available car in the map
Event flow	1. User clicks on the interested icon.

	2. User can check its status, that includes its charge, current position and address. 3. User can easily reserve the car by clicking the corresponding button. (i.e. “RESERVE CAR”)
Output condition	The user has reserved a car so the reservation timeout start.
Exception	There are no available cars in the area or the car he selected is already reserved so he can’t proceed to the reservation.



3.2.6 Delete reservation

Since a user reserved a car, he should have the possibility to decline the reservation. In this situation, application shows the user through the map the way to reach the car, and gives him/her to options: first one is to unlock the car, when he/she is nearby; the second one is to decline reservation: this option can be selected from any user’s position, since closeness is not requested.

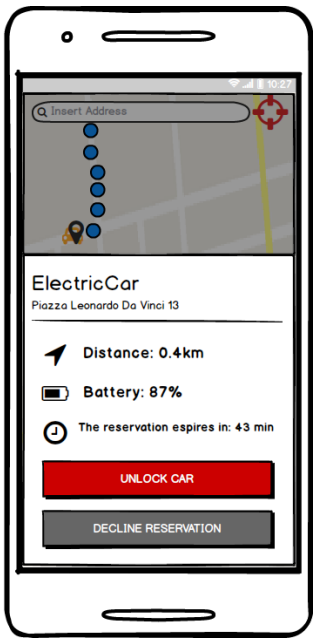
Actor	**User**
Goal	Goal n°5
Input condition	User is logged to the system and has already reserved a car.

Event flow	1. Application shows to the user status of reservation. 2. User can decline his/her reservation clicking “DECLINE RESERVATION” button.
Output condition	User is redirected to the map where he can perform another reservation.
Exception	

3.2.7 Unlock the car

Since a user reserved a car, he should be able to unlock the car, when he/she reaches it. The car’s system should be able to detect user’s position: if he/she is near enough, it should be possible to unlock the car by clicking on the related button on mobile application.

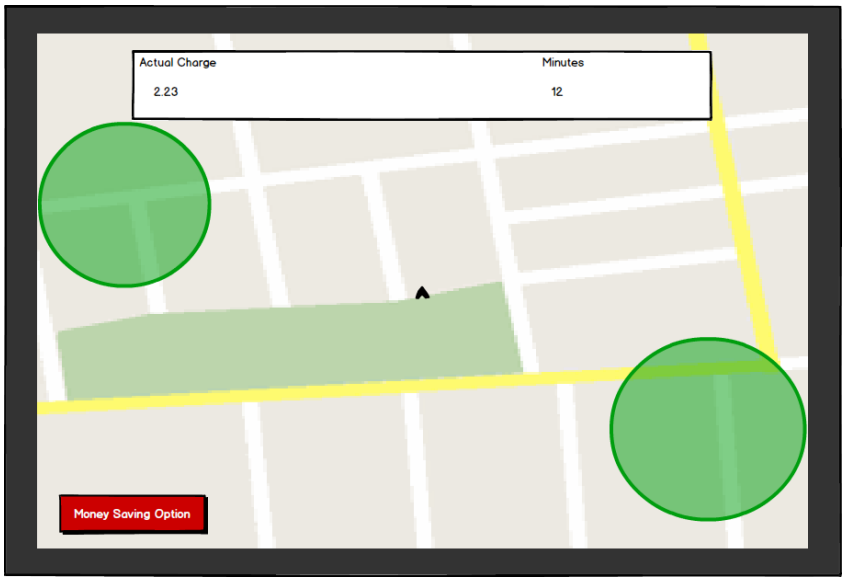
Actor	**User**
Goal	Goal n°6
Input condition	User is logged and has already reserved a car.
Event flow	1. Application shows to the user status of reservation. 2. User can unlock the car clicking “UNLOCK CAR” button (this function works if and only if the user is sufficiently near the car)
Output condition	The car is unlocked by the system so the user can enter the car and start the ride, while the reservation is marked as completed.
Exception	User is not sufficiently near the car. In that situation, the system can’t detect user closeness and unlock the car.



3.2.8 View charge during the trip

A user logged into the system, after starts his car-sharing travel, can find on the screen of the car the amount of money he will pay for the trip, instantly. So, the system update the on real time the costs of the service, also calculating the possible discount of the travel for good behaviour.

Actor	**Logged User that have rent a car**
Goal	Goal n°7
Input Condition	User must be logged and must have already picked up a car and started the rent
Event Flow	1. Car-sharing services start, and the user can begin his travel 2. Car's display shows the trip's charge
Output condition	The system provides some information about the cost of the service and about the travel, in real time
Exception	

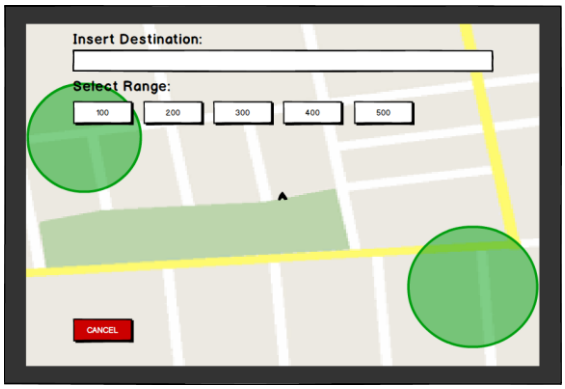


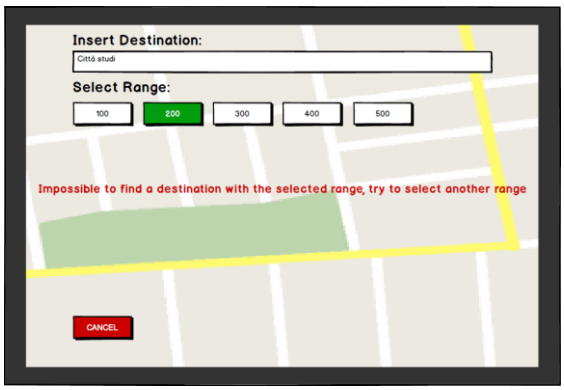
3.2.9 Enable “Money saving” option

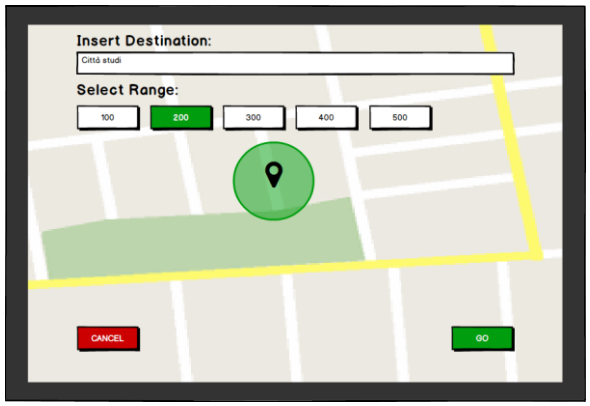
The system provides the possibility of some discount if the user has some attention for the environment. For example, it provides a discount if at the end of the rent, the user plugging in the car. This discount tempts the user, and make more probably that the next user will find a full-battery car nearby. If a lot of user do this, the car-sharing environment work well. So, when the user reach the reserved car, can input his final destination and the system provides information about the station where to leave the car to get a discount.

Actor	**Logged User**
Goal	Goal n°8
Input Condition	The user must be logged and must have already picked up a car
Event Flow	1. User before starting the rent can enable “money saving” option 2. The system asks for a final destination

	3. The system also asks for a range from the destination where it must search for the final position 4. The system display advice in order to leave the car in a safe area where plugs are available
Output condition	The system provides information about the station where to leave the car to get a discount, near the destination inserted into the system by user.
Exception	There is no final destination near the destination of the user with the selected range, so if the user want to use the money saving option he must insert a wider range.







3.2.10 Plug-in the car to get the discount

At the end of the travel, the user has some possibility to conclude his rent: he can reach a safe area, where it's possible to plug-in the car and get a discount on the service thanks to good environmental behaviour, or alternatively he can interrupt the services without

any precaution. The system give some advice to the driver, in order to increase the possibility that the cars are parked in a safe area and plugged in. This method can help the environment of the service because if a lot of car are plugged in, there are more possibility that some users will find an available car when it is requested.

Actor	**Logged User that have rent a car**
Goal	Goal n°8
Input Condition	The user must have leaved the car into a charging station and pressed “stop” button on car’s display to communicate to the system that the rent is finished.
Event Flow	1. User should leave the car 2. User should plug-in the car
Output condition	The system recognise that the rent is stopped and the car is plugged in so it applies a certain discount to the user
Exception	The system won’t apply the discount if the user doesn’t plug the car.

3.2.11 Visualize “trip review”

At the end of the trip, when the user select the stop button to stop the travel, on the screen of the car and of the smartphone appear the “trip review” associated to the last ride. This provides information related to length of the trip in terms of minutes, the different discounts that are applied on the cost of the travel and the overall cost.

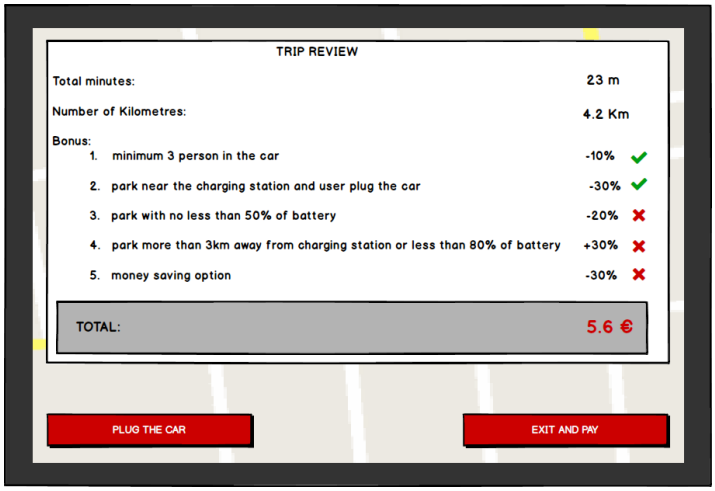
Actor	**Logged User that have rent a car**
Goal	Goal n° 8,9
Input Condition	The user must have rent a car and started the trip.
Event Flow	1. During the trip with the car, user can press the stop button in order to stop the trip;

	2. The system at this point shows the “trip review” associated to the last ride with the car.
Output condition	The system provides information related to the trip, so the user get information related to the last ride and the discount that he had achieved.
Exception	Exception occurs if the user press the button in areas that are not safe, in this case he cannot stop the ride because he must park in safe areas

3.2.12 Conclude the rent and pay

The user, after reviewing the trip information, if there is possibility can plug the car to get another discount. Although this possibility the user after the trip review must conclude the rent with the related button and exit the car so he can finalize the payment and the rent. At this point the car become available again for the other users of the system.

Actor	**Logged User that have stop the trip**
Goal	Goal n° 8,9
Input Condition	The user must have pressed the stop button.
Event Flow	1. The user concludes the rent with the related button to pay for the trip and exit the car.
Output condition	The bill associated to the trip is paid by the user, and the car associated to that user becomes free again. So, it can be rented by other users of the system.
Exception	If there weren’t exception after the stop button then there cannot be exception at this point, except for problem related to the payment from the credit card of the user.



3.3 SCENARIOS

3.3.1 SCENARIO 1: Sign in to the system

Luca just moved to Milan from Modena. He is attending Politecnico di Milano and he lives near university, so he doesn't need annual subscription to public transport, because he walks to university. Anyway, he likes going out for dinner with friends during the week and the weekend, too. So, he wants to find an easy and cheap way to reach his preferred bars and pubs. He finds out PowerEnjoy application, which perfectly deals with his needs. He indeed decides to sign up to the system, inserting his data through mobile application, even if this operation was available on web browser, too. Registration process is quite easy: it consists in just inserting some personal information, ID's number and driving license number. Now Luca can log into the system and reserve the nearest car to start his bars' tour!

3.3.2 SCENARIO 2: Discount because of 3 persons

Simone lives in Milan with his girlfriend Giulia. They have planned an amazing trip to Turkey, booking a flight which departs from Milano Linate airport. They are good students, so they are studying at Bicocca's library and they lose the sense of time. When they finally recover, they understand they are in a terrible delay. Because the way to airport by train or bus is too long, they asked to Gian, a classmate, to bring them there. Gian has no car, so he decides to reserve an electric car provided by PowerEnjoy so he

can help their friend. He takes the car to pick up Simone and Giulia and bring them to airport, so they can get the flight in time. Moreover, because there are two passengers on the car, according to PowerEnjoy's policy which encourages picking up friends, the price for the ride to the airport is discounted.

3.3.3 SCENARIO 3: Decline reservation within one hour

Simone and his girlfriend Giulia live in Milan. Since they recently move from Bologna, they would like to visit Corso Como and Gae Aulenti square. They decide to reserve an electric car through PowerEnjoy mobile application: they can easily find one available car near their house. So, they start getting dressed and preparing stuff for the visit, when suddenly it starts raining with strong wind. This is a bad news for the guys who can't go out anymore: they can however decline their reservation for the car, since it lasts for one hour.

3.3.4 SCENARIO 4: Reservation and fee

It's midday, and Luca is planning a trip into the city centre after work, but he lived far away from the nearest station of the underground service, so he decides to use the car sharing service advised by one of his friends. Luca is logged into the system, and he can research a car: he knows that at the time he wants to start the service, he will be at home, so he starts with the reservation of a car nearby his home's position. Unfortunately, Luca cannot leave his work until seven o'clock, and he doesn't read the rules of the service so during the afternoon the reservation expired and Luca's credit card is charged of 1€ coherently with the rules of the service.

3.3.5 SCENARIO 5: Versatility of the service and discounts

During the morning, Luca have planned to reach his university, but he is in a hurry because one of his professor is waiting for a meeting. So, he reserved a car and, in time, he reaches the correct position, where the mobile application of Power Enjoy indicates there is the specific car he has reserved few times ago. As soon as he arrived nearby the car, he opens the mobile application and in the specific section, he spotted the button for unlock the car and start the rent. So, with the mobile connection enable, he decides to press the button and unlock the car. Luca is surprised by the speed of the system, that

can unlock the car almost in real time. He can enter the car and start the service in time, so he can reach his professor without any problem. Also at the end of the trip he gets a discount of 20% because he left the car with more than 50% of battery remaining.

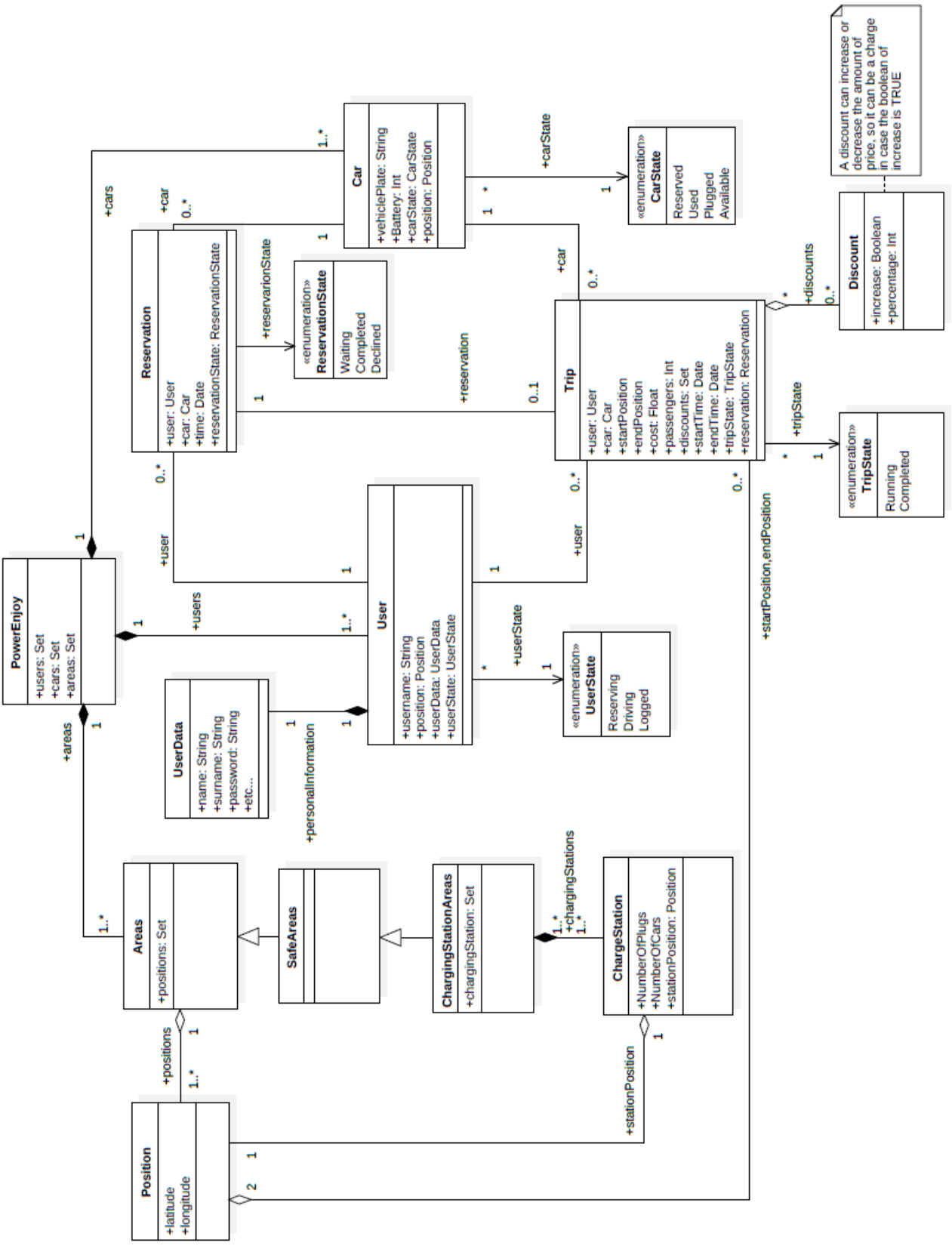
3.3.6 SCENARIO 6: Charge because of low battery level

Gian decides to use the car sharing service offered by PowerEnjoy because his parents come to visit him by train so they don't have the car and because at the same time they don't want to use public service. So, he reserves a car and they go for a trip around Milan. Since it is the first time that Gian uses this service he didn't know the policy of the service and absently he leaves the car in a safe area with less than 20% of battery. The system ad this point charges the trip of Gian by 30% because he didn't behave well from the system point of view. In fact, a user who see a car with less than 20% of battery remaining is discourage to reserve that, so the system lost money because that car is unused.

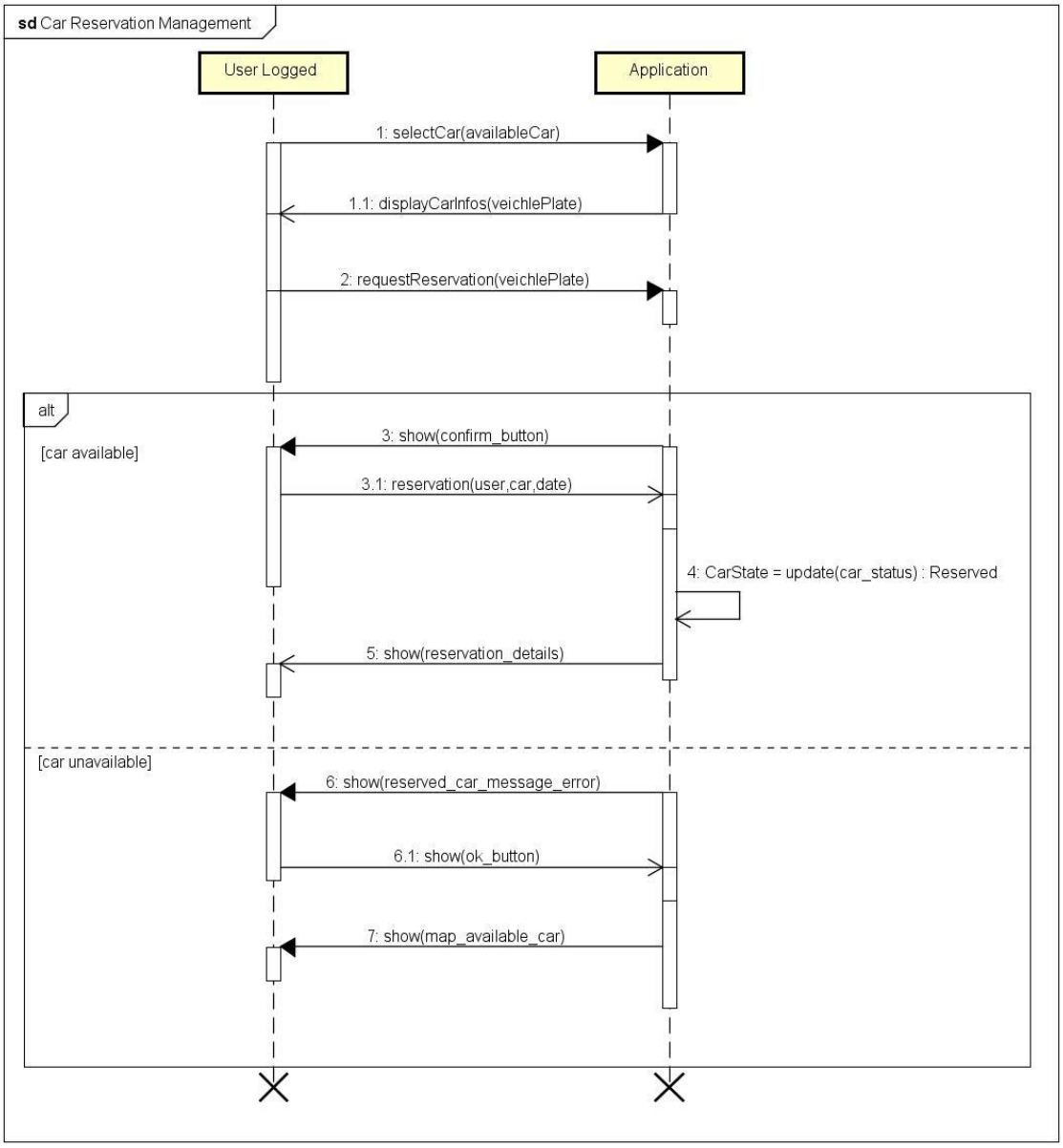
3.3.7 SCENARIO 7: Money Saving Option

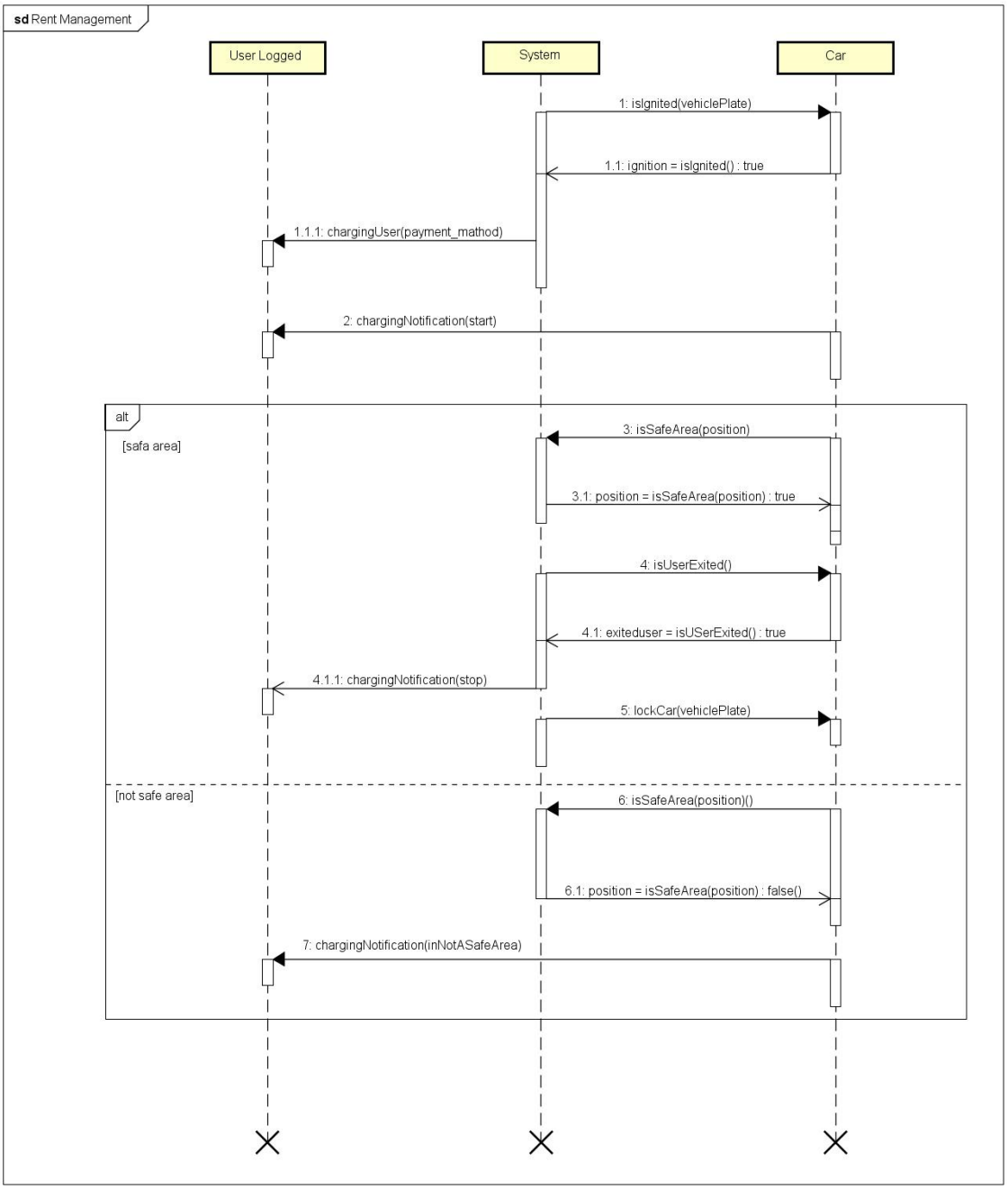
Gian is a studying at Politecnico di Milano, since he is an offsite student he must pay attention at spending money. At the same time, he has planned to exit on Saturday evening with his friend, since it's midnight and the subway is closed, he decided to use PowerEnjoy's car sharing system. So, he reserves a car and get into that after he reaches it, to save some money for the trip he decides to use the money saving option that the system offers. He selects the destination of his travel and the system provides a destination near the first one where he must leave the car. He drives to the destination and before leaving the car he takes care of plug it to the charging station. So, at the end he reaches his destination and he also gets a discount of 30% on the last trip since he activates the money saving option and moreover he contributes to the management of the system.

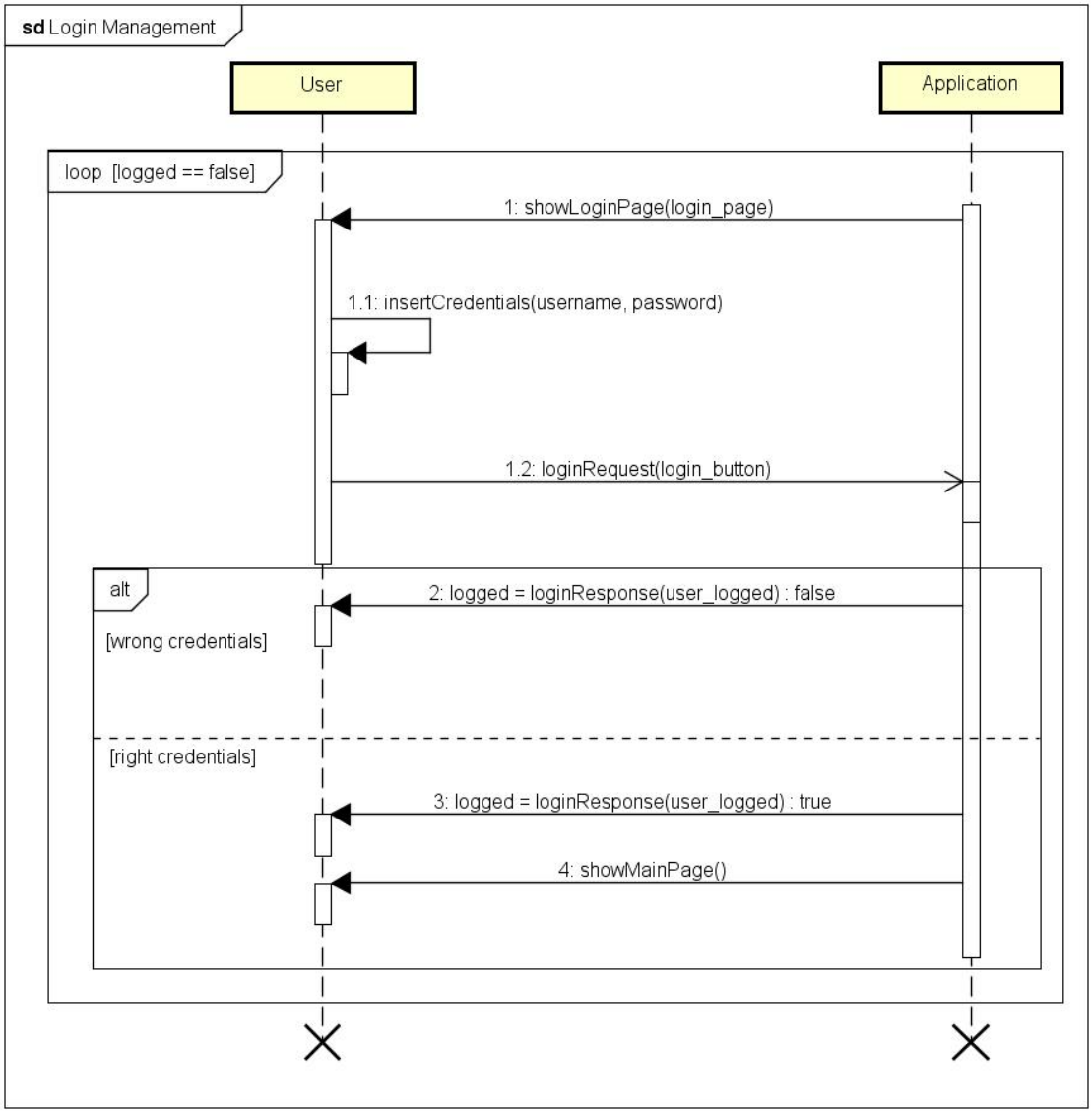
3.4 CLASS DIAGRAM



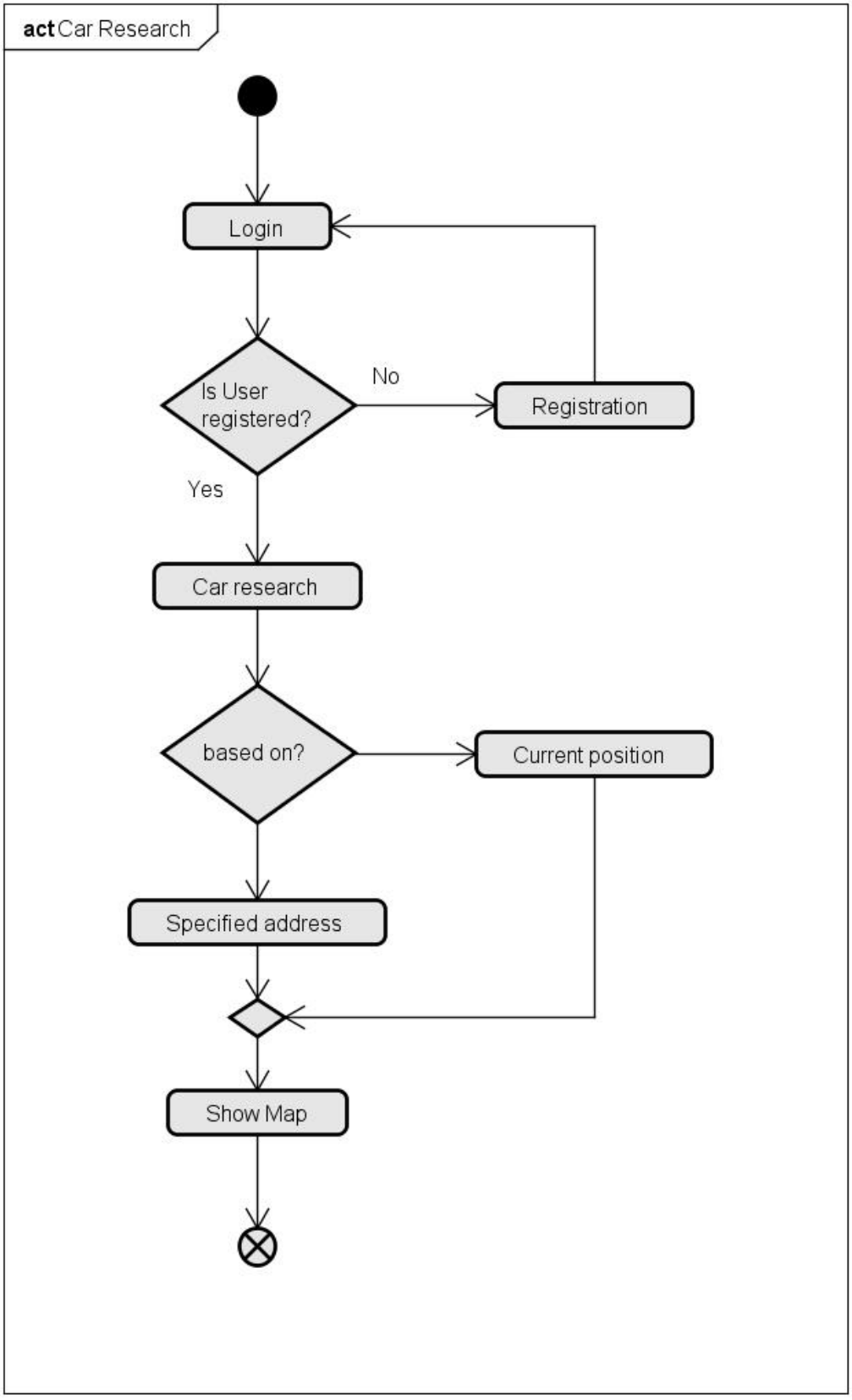
3.5 SEQUENCE DIAGRAMS

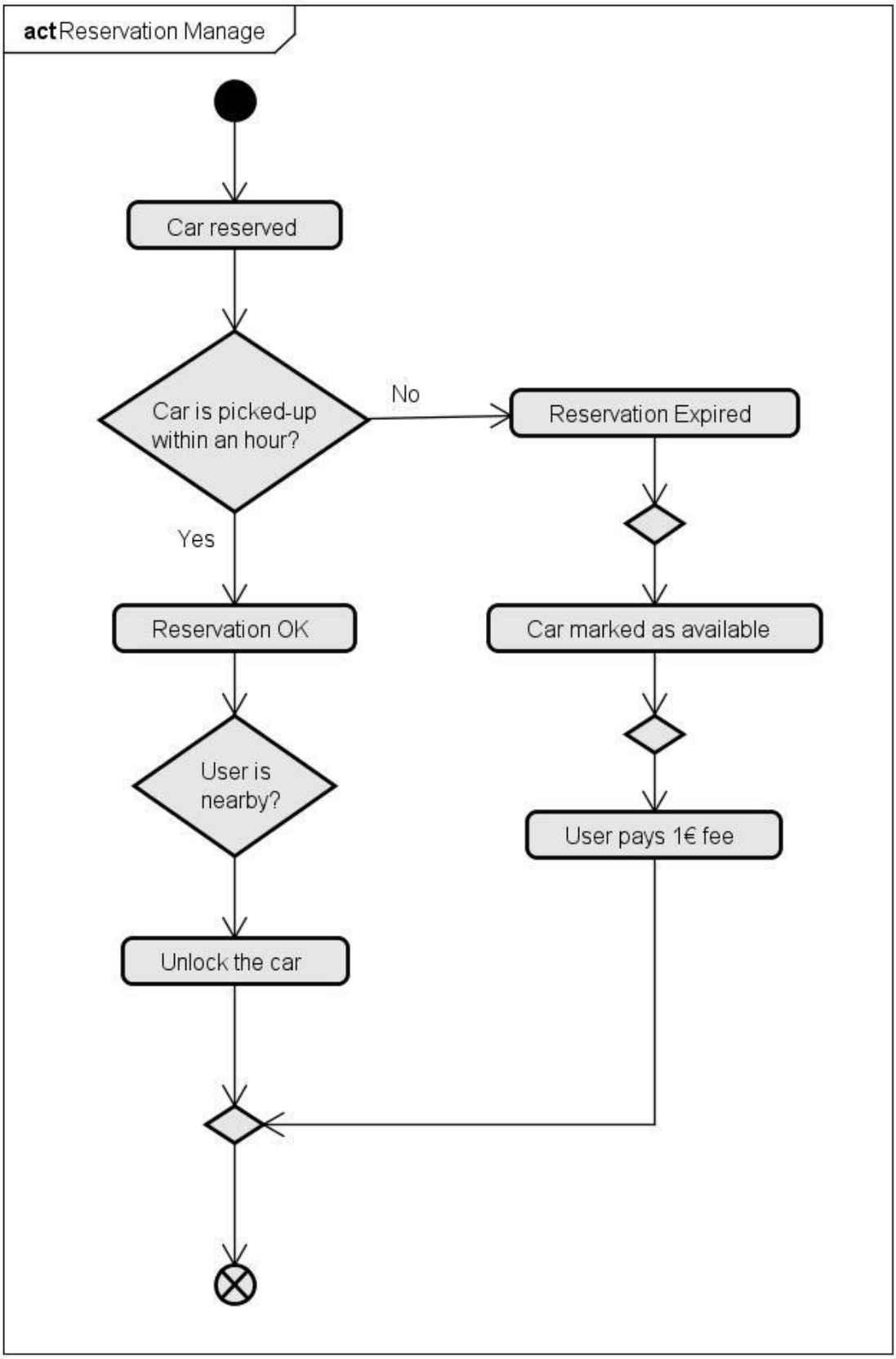



sd Rent Management
User Logged
System
Car
1: isIgnited(vehiclePlate)
1.1: ignition = isIgnited() : true
1.1.1: chargingUser(payment_mathod)
2: chargingNotification(start)
alt
[safa area]
3: isSafeArea(position)
3.1: position = isSafeArea(position) : true
4: isUserExited()
4.1: exiteduser = isUSerExited() : true
4.1.1: chargingNotification(stop)
5: lockCar(vehiclePlate)
[not safe area]
6: isSafeArea(position)()
6.1: position = isSafeArea(position) : false()
7: chargingNotification(inNotASafeArea)

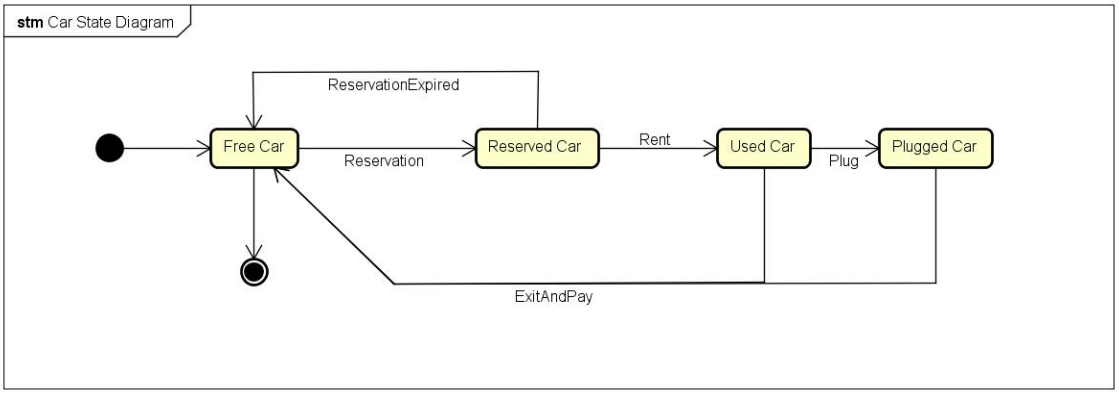

sd Login Management
User
Application
loop [logged == false]
1: showLoginPage(login_page)
1.1: insertCredentials(username, password)
1.2: loginRequest(login_button)
alt
[wrong credentials]
2: logged = loginResponse(user_logged) : false
[right credentials]
3: logged = loginResponse(user_logged) : true
4: showMainPage()

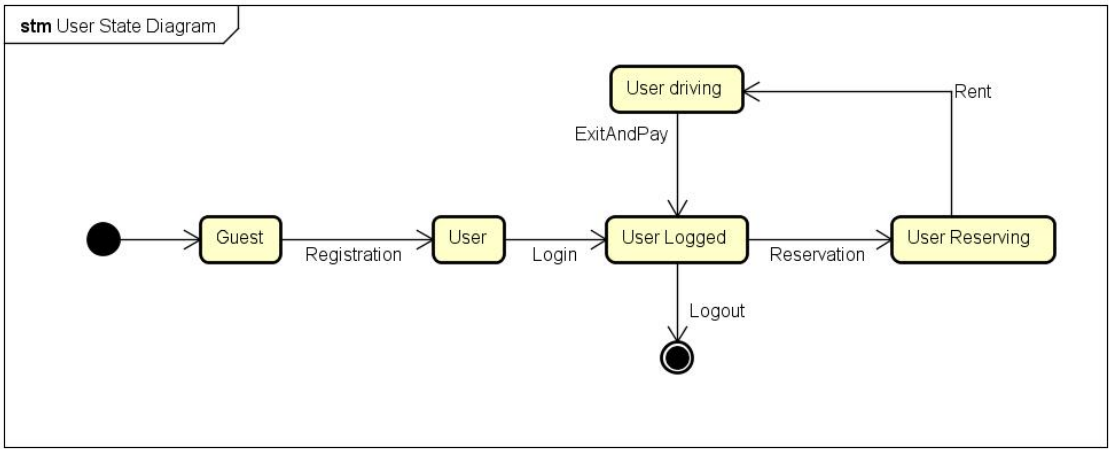
3.6 ACTIVITY DIAGRAMS




act Reservation Manage
Car reserved
Car is picked-up within an hour?
No
Reservation Expired
Yes
Reservation OK
Car marked as available
User is nearby?
User pays 1€ fee
Unlock the car

3.7 STATE DIAGRAMS





3.8 SOFTWARE SYSTEM REQUIREMENTS

3.8.1 Availability

To guarantee the maximum profit by the service, the system must be available 24 hours per day and 7 day per week. The system insures a minimum availability of 97%. 3% of time is removed away from availability to provide the possibility of some update during the year.

3.8.2 Security

To guarantee complete security about user's data, the whole database is under specific cryptography, and the system's administrators cannot access into the personal data of the users. Password, credit card's code and personal data are visible only for the owner. All the access into the database are recorded and saved in a specific log file, in order to keep tracked and controller database's accesses. Moreover, data from different sector, like personal data related to some travel and personal data related to credit card, are saved and used in 2 different part of the system, to guarantee the security about the use of this kind of private data.

3.8.3 Maintainability

The system is backed up 3 times per day, in order to guarantee the database integrity and consistency. Loss of data is not permitted, so the database is saved in 2 copy and the power supply is insured by a special system that keep online at least one database.

3.8.4 Portability

The software is designed with the aim of easy installation, because it is developed in Java, so all the computer with Java installed can execute correctly the application. Mobile phones with Android operating system are able to execute mobile application, personal computer with a modern browser are able to display the webpage of the application. The database is created with the most common platform, like SQL, so the maintainer can port database from one support to another easily.

3.9 ALLOY

3.9.1 Alloy Modelling

We report below the code modelled in Alloy, using screenshots with colours and indentation to favour the reading and the comprehension.

We inserted three assertions to verify is our model could be valid.

We use predicates to construct different world to show at the best the peculiarities of our model.

```
/*==================ENUMERATIONS==================*/

enum ReservationStatus {
   WAITING,
   COMPLETED
}

enum CarState{
   RESERVED,
   USED,
   FREE
}

enum TripState {
   RUNNING,
   ENDED
}

enum UserState {
   RESERVING,
   DRIVING,
   LOGGED
}

/*==================SIGNATURES==================*/

one sig PowerEnjoy{
   areas : set SafeArea,
   chargingStations : set ChargingStation,
   users: set User,
   cars: set Car
}

sig Position {}

abstract sig Area{
   position : one Position
}

sig SafeArea extends Area {}

sig ChargingStationArea extends SafeArea {
   chargingStation: one ChargingStation
}

sig ChargingStation {
   position: one Position,
   plugs: Int,
   cars: set Car
}{plugs > 0 and (#cars <= plugs) and (#cars > 0) and (cars.position = position) }

sig Car{
```

```
sig Car{
// ID: Int,
   batteryPercentage: Int,
   plugged: Bool,
   position: one Position,
   state: one CarState
}{(batteryPercentage > 10) and (batteryPercentage < 100)}

sig User{
   ID: one Int,
   position: one Position,
   state: one UserState
// personalInfo : one PersonalInformation
}{ID > 0}

sig PersonalInformation {}

sig Reservation{
   user: one User,
   car: one Car,
   date: one Date,
   time: one Time,
   status: one ReservationStatus
}

sig Date{}

sig Time{}

/*In the definition of the signature Trip we add the constraints related to the number of passengers,
the discounts that can be applyed to the trip and the relation between the trip and the reservation associated.*/
sig Trip {
   startPosition: one Position,
   finalPosition: lone Position,
   user: one User,
   car: one Car,
   date: one Date,
   startTime: one Time,
   endTime: lone Time,
   discount: set Discount,
   reservation: one Reservation,
   passengers: Int,
   state: one TripState,
   moneySavingOption: one Bool
}{ (startPosition != finalPosition or finalPosition = none)
      and (startTime != endTime or endTime = none)
      and passengers >= 0 and passengers <= 4
      and (reservation.car = car) and (reservation.user = user)
      and (reservation.date = date)
      and ((passengers >= 2) iff ( Discount_10_OFF in discount))
      and (( Discount_10_OFF in discount) iff (passengers >= 2) )
      and ((Discount_30_OFF in discount) iff ((one c: ChargingStation | finalPosition=c.position) and (car.plugged=True)
                                              and car.position = finalPosition))
      and ((Discount_20_OFF  in discount) iff (car.batteryPercentage >= 50))
      and ((PenaltyCharge_30_UP in discount) iff (car.batteryPercentage <20) )
      and ((car.batteryPercentage <20) iff (PenaltyCharge_30_UP in discount))
}
```

```
abstract sig Discount {}

lone sig Discount_10_OFF extends Discount{}

lone sig Discount_20_OFF extends Discount{}

lone sig Discount_30_OFF extends Discount{}

lone sig PenaltyCharge_30_UP extends Discount{}

/*==================FACTS==================*/

/*Users,cars,charging stations and safe areas belongs to PowerEnjoy system*/
fact AllUsersInSystem {
   all u: User | u in PowerEnjoy.users
}

fact AllCarsInSystem {
   all c: Car | c in PowerEnjoy.cars
}

fact AllChargingStationsInSystem {
   all c: ChargingStation | c in PowerEnjoy.chargingStations
}

fact AllSafeAreasInSystem {
   all s: SafeArea | s in PowerEnjoy.areas
}

/* No discount can be applied until the trip is ended*/
fact noDiscountUntilEnded {
   no t: Trip | ((t.discount != none) and (t.state = RUNNING ))
}

/*There not exist two charging station with the same position*/
fact NoChargingStationWithSamePosition {
   no disjoint c1,c2: ChargingStation | c1.position = c2.position
}

/*There not exist two cars with the same position*/
fact NoCarsWithSamePosition {
   no disjoint c1,c2: Car | c1.position = c2.position
}

/*No two reservation on the same car*/
fact noSameCarsReseved{
   no disjoint r1,r2: Reservation | r1.car = r2.car
}

/*One reservation belongs only to one trip*/
fact noTripWithSameReservation {
   no disjoint t1,t2 : Trip | t1.reservation=t2.reservation
}
```

```
/*Constraints regarding the status of the reservation*/
fact ReservationStatus{
   all r: Reservation | (r.status = COMPLETED) iff ( one t: Trip | t.reservation = r)

}

fact ReservationWaiting {
   all r: Reservation | ( (r.status = WAITING) iff ( no t: Trip | t.reservation = r) )
                     and   (( no t: Trip | t.reservation = r) iff (r.status = WAITING) )
 }

/*Constraints regarding the state of the car*/
fact CarState{
   all c: Car | (c.state = RESERVED) iff (one r: Reservation | ((r.car = c) and (r.status = WAITING)))

   all c: Car | (c.state = USED) iff (one t: Trip | ((t.car = c)))
}


/*Constraints regarding the state of the user*/
fact UserState{
   all u: User | (u.state = RESERVING) iff (one r: Reservation | ((r.user = u) and (r.status = WAITING)))

   all u: User | (u.state = DRIVING) iff (one t: Trip | ((t.user = u) and (t.state = RUNNING)))
}

/*Constraints regarding the state of the trip*/
fact TripState {
   all t: Trip | ((t.state = RUNNING) and (t.endTime = none) and (t.finalPosition = none)) or
          ((t.state = ENDED) and (t.endTime != none) and (t.finalPosition != none))
}

/*No user can reserve a car if he is already reserving one*/
fact noUserReserveTwoCars{
   no disjoint r1, r2: Reservation | r1.user=r2.user
}


/*Charging station must be in a position which belongs to chargingStationArea*/
fact ChargingStationInChargingStationArea {
   all c: ChargingStation | c.position in ChargingStationArea.position
}

/*Every charging station belong only to a single charging station area*/
fact NoChargingStationInDifferentChargingStationArea {
   no disjoint a1, a2: ChargingStationArea | a1.chargingStation = a2.chargingStation
}

/*MoneySavingOptionConstraint*/
fact MoneySavingOptionEnabledImpliesFinalPosition{
   all t: Trip | t.moneySavingOption=True iff (t.finalPosition!=none)
}
```

```
/*==================ASSERTIONS==================*/

assert noCarIsReservedFromTwoReservation {
   no disjoint r1, r2: Reservation | r1.car = r2.car
}

check noCarIsReservedFromTwoReservation

assert noUserCanReserveMoreThanOneCar{
   no disjoint r1, r2: Reservation | r1.user = r2.user
}

check noUserCanReserveMoreThanOneCar

assert noUserCanUseMoreThanOneCar{
   no disjoint t1, t2: Trip | t1.user = t2.user
}

check noUserCanUseMoreThanOneCar

assert NoMoneySavingOptionActivatedWithoutFinalPosition {
   no t: Trip | t.moneySavingOption = True and t.finalPosition = none
}

check NoMoneySavingOptionActivatedWithoutFinalPosition

/*==================PREDICATES==================*/
pred showPenaltyCharge(){
  #ChargingStation != 0
   some t: Trip, p: PenaltyCharge_30_UP | p in t.discount and t.car.batteryPercentage > 10
}

pred showDiscounts(){
   #ChargingStation != 0
   #Trip != 0
   #Reservation != 0
   some t: Trip, p: Discount_20_OFF,  p1: Discount_10_OFF, p2: Discount_30_OFF|
   p in t.discount and p1 in t.discount and p2 in t.discount
}

pred showWaitingReservation(){
   one r: Reservation | r.status = WAITING
   some t: Trip, p: Discount_20_OFF,  p1: Discount_10_OFF, p2: Discount_30_OFF|
   p in t.discount and p1 in t.discount and p2 in t.discount
}



run showPenaltyCharge for 8 Int
run showDiscounts for 8 int
run showWaitingReservation for 8 Int
```

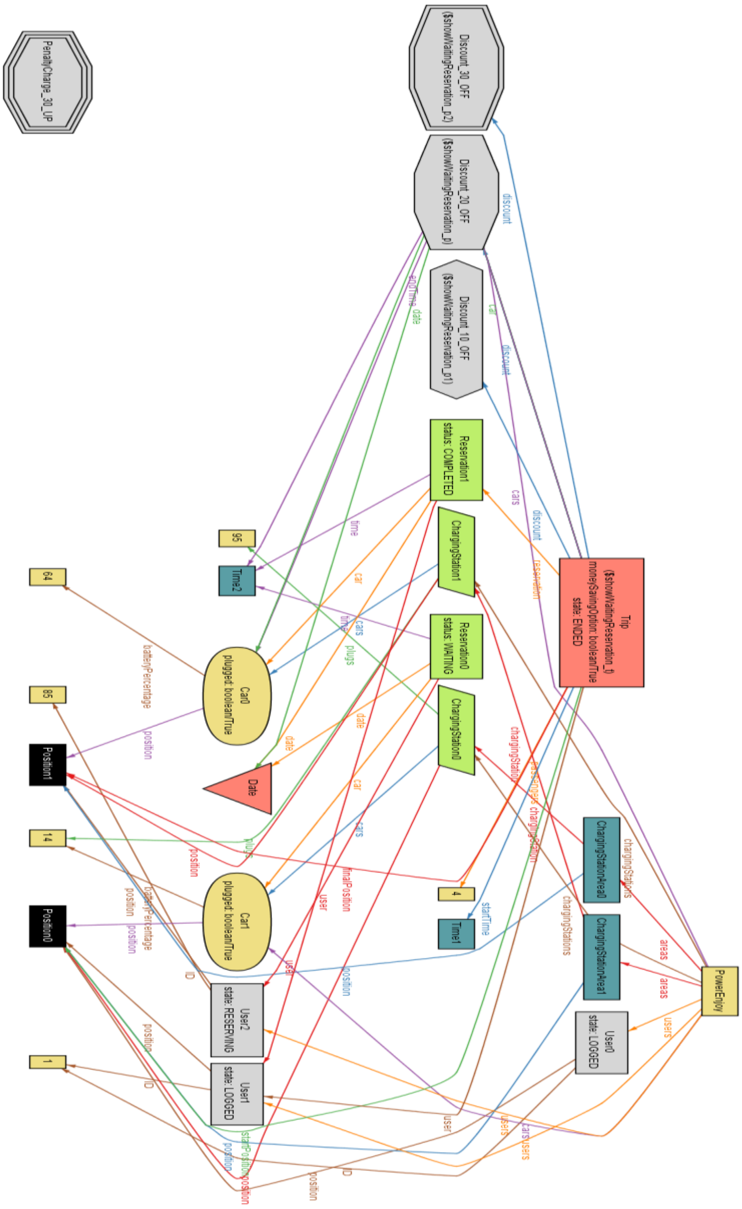
3.9.2 Results

Here is the report of Alloy Analizer.

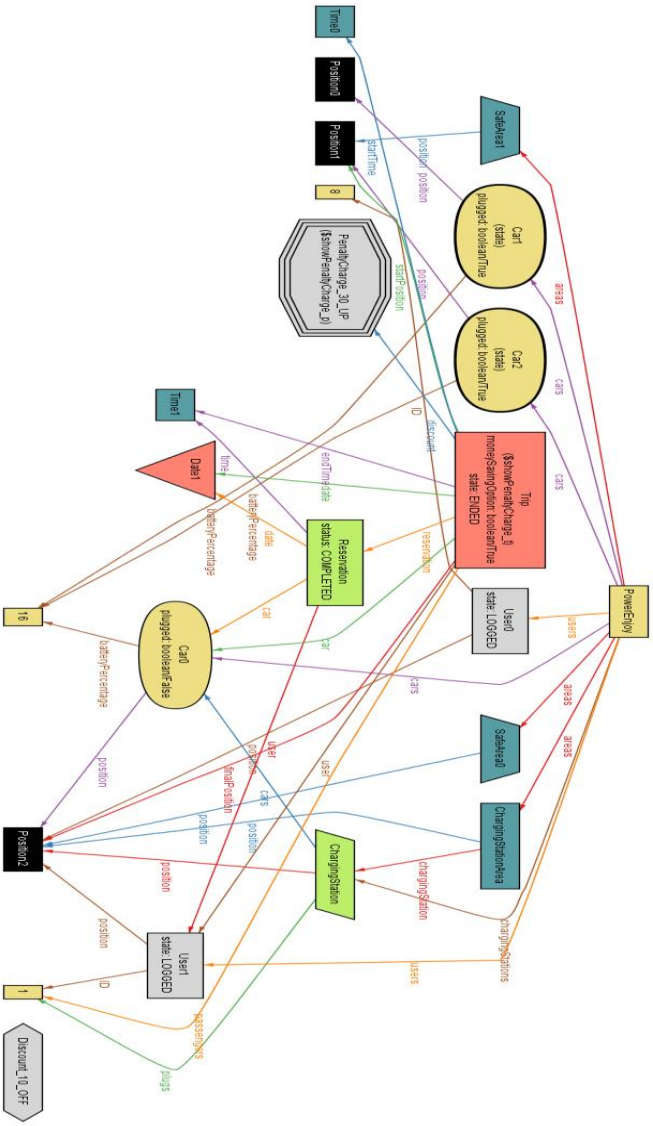
7 commands were executed. The results are:
#1: No counterexample found. noCarIsReservedFromTwoReservation may be valid.
#2: No counterexample found. noUserCanReserveMoreThanOneCar may be valid.
#3: No counterexample found. noUserCanUseMoreThanOneCar may be valid.
#4: No counterexample found. NoMoneySavingOptionActivatedWithoutFinalPosition may be valid.
#5: **Instance found.** showPenaltyCharge is consistent.
#6: **Instance found.** showDiscounts is consistent.
#7: **Instance found.** showWaitingReservation is consistent.

3.9.3 World Generated

In the first picture, we want to show a world where a reservation not in association with a trip has status WAITING; the user who is waiting for this reservation is in state RESERVING. We associate a trip to three different type of discounts and to one penalty charge: we apply 10% discount if the trip has 2 or more passengers; then we apply 20% of discount if the car is left with more than 50& of battery and last we apply 30% off if the car is plugged (plugged=true) and it's left in the same position of a charging station area (this means it's in charge). Moreover, we apply a 30% penalty charge if the car is left with less than 20% of battery. Notice that a discount is applied only to a ENDED-state trip, which matches discount's sufficiency conditions. User's state must be logged to reserve and use a car.


PenaltyCharge_30_UP
Discount_30_OFF ($showWaitingReservation_p2)
Discount_20_OFF ($showWaitingReservation_p)
Discount_10_OFF ($showWaitingReservation_p1)
Reservation1 status: COMPLETED
ChargingStation1
Reservation0 status: WAITING
ChargingStation0
Trip ($showWaitingReservation_t) moneySavingOption: booleanTrue state: ENDED
Time2
Time1
Car0 plugged: booleanTrue
Car1 plugged: booleanTrue
Date
Position1
Position0
User2 state: RESERVING
User1 state: LOGGED
User0 state: LOGGED
ChargingStationArea0
ChargingStationArea1
PowerEnjoy
95
64
85
14
4
1

In this second world, we show the penalty charge.



4 BIBLIOGRAPHY AND DETAILS

4.1 TOOLS USED

- **Microsoft Office Word**: to write this document.
- **GitHub**: to create a repository and share our work.
- **Astah**: to create UML diagrams.
- **Balsamiq**: to create mock-ups.
- **StarUML**: to create class diagram.
- **Alloy Analyzer**: to simulate specifications and perform validation of our system.

4.2 HOURS OF WORK

- Simone Bruzzechesse: 32 hours
- Luca Franceschetti: 32 hours
- Gian Giacomo Gatti: 32 hours

4.3 BIBLIOGRAPHY