POLITECNICO
MILANO 1863

13 November 2016

PowerEnJoy

Requirements Analysis and Specifications Document

Blanco	Federica	875487
Casasopra	Fabiola	864412

Software Engineering 2 Project
2016/2017

Contents

List of Figures

List of Tables

1 Introduction

The main purpose of this document, the project's scope and its actors and goals are described in this section. Moreover, we are going to give some definitions which will help the reader to understand the content of this document and we are going to show the reference documents that have been used to redact this one.

1.1 Purpose

This document is called *Requirements Analysis and Specifications Document*, also known as RASD. Its purpose is to communicate to customers what is needed to understand functional and non-functional requirements, the limitations and obstacles for implementing this car-sharing system and the constraints of this specific problem. In addition, the RASD is useful in order to figure out how to model the customers' needs. This document is also addressed to developers and programmers who will have to implement all the requirements. For this reason, it must be complete and correct as much as possible. The RASD is a contract with customers, therefore it must show use cases to allow everyone to understand what the system will do and in what domain it can be used. Besides, the project manager can use this document to evaluate the costs and size of the project.

1.2 Scope

The aim of the project PowerEnJoy is to provide a car-sharing service that involves *only* electric cars. All people who want to rent a car must be able to register to the system using credentials such as name, surname, e-mail, nickname and other anagraphic data. Moreover, they have to give valid driving licence information and valid payment information (i.e. number of credit card), that is needed for the payment of the service. When a user receives the password to log in, he can find available cars in a specific location and, if he wants, he can reserve one of them. The system unlocks the car as soon as the user is nearby and keep him informed about the amount to pay for the service with a screen on the car; when the car is in one of the safe areas, indicated in a list that can be consulted on-line, the user can park. Once he exits from the car, the system locks it. The project has also the purpose to encouraged people to left at home their pollutants cars and take the electric car with other people: in fact, if there are at least two passengers with the driver, he has a 10 per cent discount on the last ride. If the user leaves the car in the safe area with more than half of the battery, he has a 20 per cent discount on the last ride and if he also plug the car into the power grid, he has 30 per cent discount, always on the last ride. However, if the user leaves the car far away from a power grid station or with more than 80 per cent of empty battery, the system charges 30 per cent more on the last ride in order to compensate for the cost required to re-charge the car on-site.

1.3 Domain Properties

We supposed that our domain has this properties:

D1 The nickname and the e-mail are unique to identify an user.

D2 The e-mail must be syntactically correct and must correspond to an existing server.

D3 The car's position can be knowed thanks to GPS.

D4 The charge is calculated considering time passed using the car, given a specific fee per minute.

D5 In the car there is a sensor that idicates how many people are in the car and another sensor which indicates the battery level.

D6 For each car there is an insurance.

D7 Reservation's time must be included between 00:00 and 23:59.

D8 The position inserted by the user must be an existing place.

D9 All the payment functionality, such as check if the credit car is valid and make the transaction secure, are provided by a Payment Service Provider.

1.4 Actors

Here there is a list of the actors who can operate with the system.

→ VISITOR: the person who visits the systems but that is not logged-in in the site. He can only see the home page and the page with the form for the registration, where he must provide all the requested information. Moreover, he has the possibility to log-in with the password given by the system when the registration is successfully committed.

→ USER: the person who has successfully logged-in. He can do all the operations provided by the system through the user interface such as reserve a car, consult the list of available cars or say to the system that he is nearby the reserved car.

1.5 Goals

Now, we want to explain exhaustively what are our system's goal.

G1 The system must be able to allow visitor to register.

G2 The system must be able to give a password to a visitor successfully registered and allow him to modify his profile.

G3 The system must be able to allow visitor to log in.

G4 The system must be able to provide to the user the list of available cars near his position or a specific location.

G5 The system must be able to allow user to reserve a car up to one hour before it is picked up.

G6 If the user is near his reserved car the system must unlock it.

G7 The system must be able to charge the user and shows all transaction's details.

G8 The system must be able to keep informed the user about the charge trough a display in the car.

G9 The system must inform the user about the reservation timer. If the user not pick up the car after one hour from the registration, the system must be able to delete the registration giving a fee of 1 euro to the user.

G10 The system must allow user to delete a reservation before one hour is passed.

G11 If a registration is deleted the system must be able to make the car available.

G12 The system must lock a car when it is left in a safe area.

G13 The system must be able to apply discount if is verified one case.

G14 The system must be able to apply an increase to the amount of a ride if is verified a specific case.

G15 The system must show to the user a list of safe areas near a specific position.

G16 The system must show to the user a list of power grid station near a specific position.

1.6 Definitions, acronyms, abbreviations

1.6.1 Definitions

The following part is necessary for avoid ambiguity or misunderstanding during the reading of this document.

- VISITOR: he is a person that is not register in the system. He can only see the homepage and go to the form for the registration or log-in.
- USER: he is a person that is registered and logged in the system. He is identified by a name, surname, nickname, e-mail, password (given by the system at the end of the registration), telephone number, address, driving license information and all the payment information (such as number of credit card and card's deadline) that must be verified by the interaction between the system and the Payment Service Provider. He can do all the services that are provided by "*PowerEnJoy*".His position is knowed using the GPS.
- CAR: we intend an electric car that can be rent trough the system. It can be available or reserved and the system will lock and unlock it when necessary. It is parked in a safe area and it may be plugged into a power grid station.The system knows its position thanks to the use of GPS.
- RIDE: we intend all the route that a user accomplishes with the same car from the moment when he pick up the car to the moment when he left it in a safe area.
- CHARGE: we intend the debt that the user must pay at the end of a ride. It is calculate from a specific amount of money per minute: the timer starts at the begin of the ride until the end of it. The charge can be also modified by some discount or a fee when particular situations occur. It also exists a charge if the user reserve a car but he doen't pick it up whithin one hour from the reservation.

1.6.2 Acronyms

Here there is the acronims list:

RASD	Requirements Analysis and Specifications Document
UML	Unified Modeling Language
API	Application Programming Interface
DB	DataBase
OS	Operating System

1.6.3 Abbreviations

- **Gn** : indicates the goal's number
- **Rn** : indicates the requirement's number for a specific goal
- **Dn** : indicates the domain's number

1.7 Overview

This document is structured in five parts:

Section 1 There is an introduction with this document's purpose, the scope of this project and its aim, the actor that can use the system and in which way, some definitions to avoid misunderstanding during the reading of the document and, most important, the goal of our project described all in a brief but comprehensive way.

Section 2 There are more specifications about the requirements, the interfaces with external agents, constraints and assumptions.

Section 3 It is very important because there are all the models for the requirement. They are modeled using UML diagrams such as *Use Case* and *Sequence Diagram*

Section 4 There is a modelization of the problem using Alloy. Together with the UML, they are very important to understand all the functionality of the system.

Section 5 There is the Appendix with information on the tools used and the hour spent by all of us to redact this document.

1.8 Reference Documents

- Specification document: Assignments AA 2016-2017.pdf
- Standard for RASD: IEEE Std 29148-2011
- API information: `https://developers.google.com/maps/documentation/geolocation/intro`

2 Overall Description

In this section, we will explain the product prospective and major functionality and also the characteristic of the user that we think he will use this application. Moreover, we will show all the constraints we have found and all the assumptions that we have made in order to make the requirements clear in all their parts.

2.1 Product perspective

Our application is a self-contained product based on a client interface, that helps the user to interact with this system. The application also uses an API for the geolocation provided by ***Google Maps*** because we need to know both the cars and user's positions. It is supported by common browser such as "*Chrome*". It is also closely linked with a DB that contains all users and cars data, the list of safe areas and if there is a power grid station in there. Because of our assumptions, the system must collaborate with the motorization's system to check user's driving license and with an external society which provides the insurance for the cars.The system must also collaborate with a Payment Service Provider that checks all the payment informations and makes transaction secure both for the user and for the software's owner.

2.2 Product functions

2.3 User characteristics

The user we expect will use our application is a person that wants to rent a car near his position, whenever he needs, in an efficient and rapid way; he is also a person who is interested in the problem of pollution since our system involves only electric car. Our application doesn't require a particular ability in using IT equipment: in fact, thanks to the friendly user interface, he must only be able to do simple actions, such as insert his credential for registration, log in and eventually set the position from where he wants to share the car. Thanks to the position, the system will do all the operation to lock and unlock the car and charge the user.

2.4 Constraints

Our software must be used at the same time by a lot of users. It doesn't have to consume a lot of battery and memory since it will be used also on mobile.

For the complete development and functionality of our product, we need to know the position of the user. In order to do that, the user, when registering at our service, must agree with our policy of privacy; otherwise, he won't be able to use our software. Another constraint is that, since we use a Google API, we must follow its regulation and developer guide.We must also follow the PCI Security Standard Council because we menage payment actions.

2.5 Assumptions and Dependencies

1. There isn't a privileged user: all the users can do the same things and there isn't a limitation in system's use.
2. The visitors can only see the homepage, the registration's form and the log in page.
3. There isn't dependency from users.
4. A user can't reserve two cars at the same hour at the same day: he can reserve only one car each time.
5. There is a user's page with the payment history: so, the user can see all details for a specific payment because all the operation must be clear. This is a warranty of correctness between system's owner and users.
6. When a reservation is deleted because one hour is passed, the user is informed trough a notification like this: *"The reservation for car X at the hour Y for the day Z is deleted because is passed one hour. You must pay a fee of 1 euro"*.
7. In the list of available cars is specified the car model with all the details so the user can select the best one for his needs. It is also indicated whether the vehicle is suitable for the transport of disabled persons.
8. During the reservation there is a remainder to know when the reservation will expire.
9. In the site there is a list of possible discount and its details.
10. During the registration we ask the user to insert his driving license's number: in this way the system can verify if it is valid or not.
11. The system provides the opportunity to receive trough the e-mail the password, if the user doesn't remember it.
12. The car's insurance is provided by an external society whereby we have stipulate a contract.
13. All the payment actions are menaged by an external Payment Service Provider.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interface

In this section, we will present some mockup to make the reader better understand the idea of the structure of the final web page and application.

Log In

The mockup in Figure 1 shows the login page of PowerEnJoy. Here, a VISITOR can log in into the application.

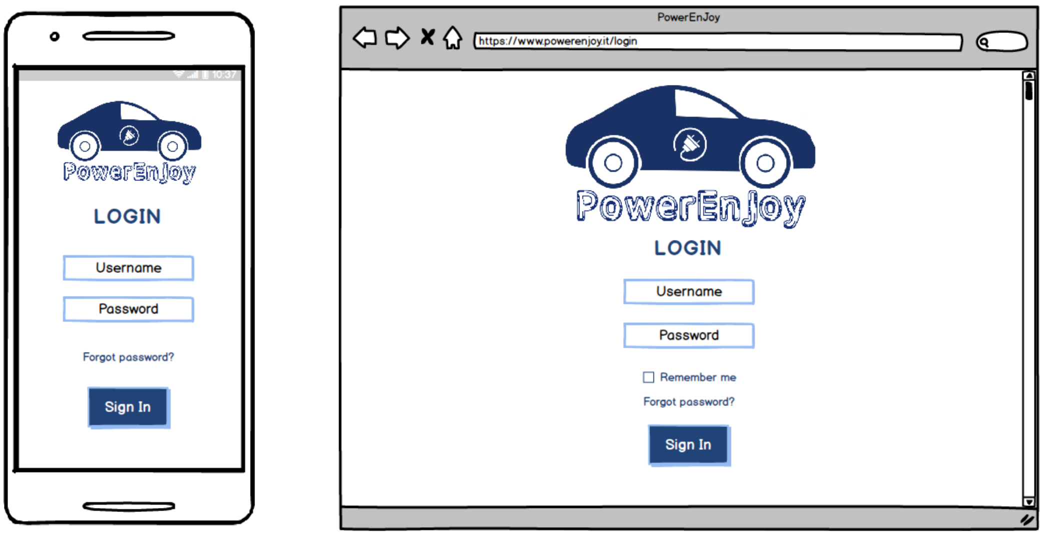


Figure 1: Log In Page mockup

Registration Form

The mockup in Figure 2 shows the registration form, where a VISITOR can sign up to access the service.

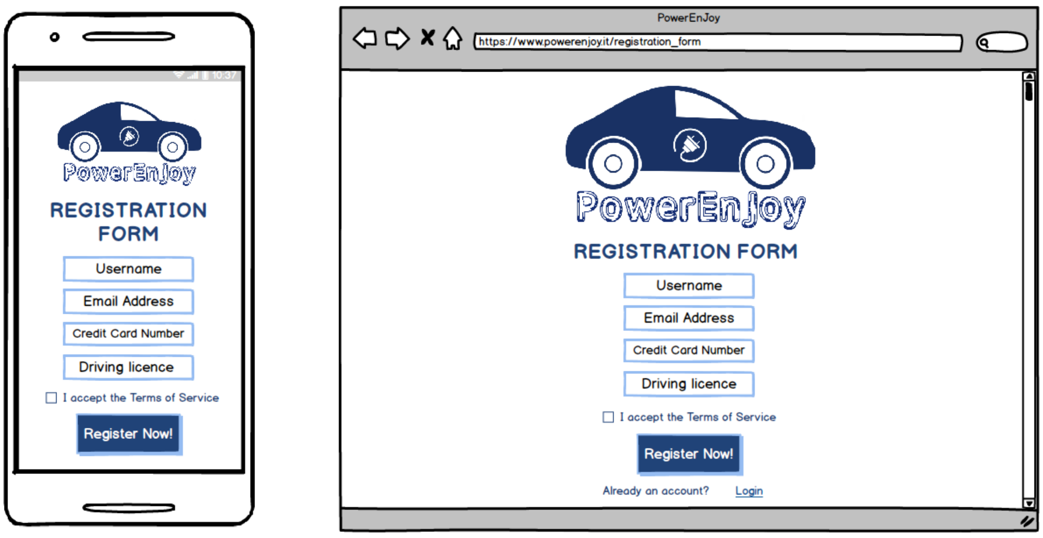


Figure 2: Registration Form mockup

Personal page

The mockup in Figure 3 shows the personal page of a USER, from where he can update his data, such as, for example, the profile picture, the password and the payment information.

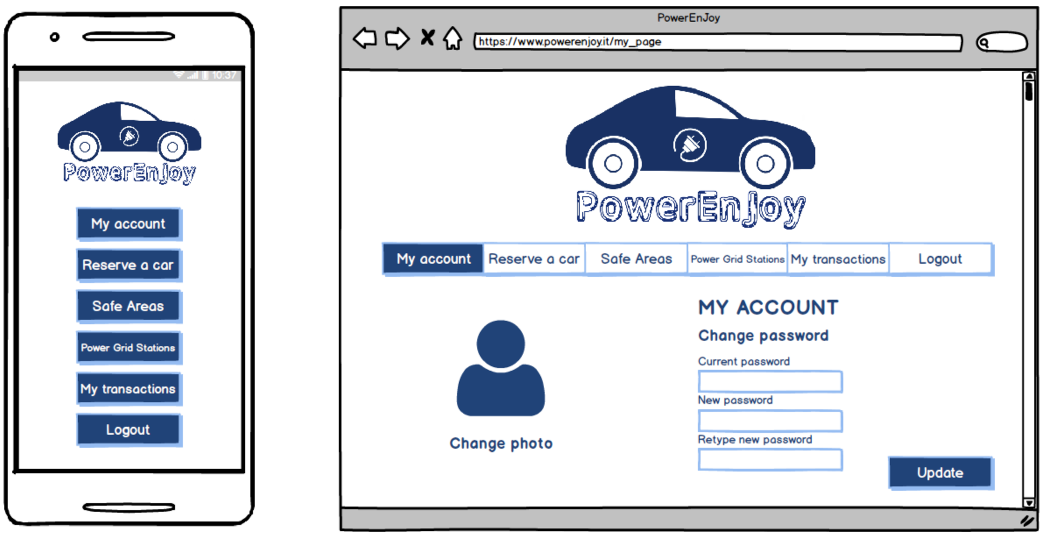


Figure 3: Personal Page mockup

Reservation page

The mockup in Figure 4 shows the page from which a USER can reserve a car nearby him or near to a selected address for an hour. He can also see how much time takes to get from the selected point to the car and its condition: level of charge and if it is plugged in a power grid station.

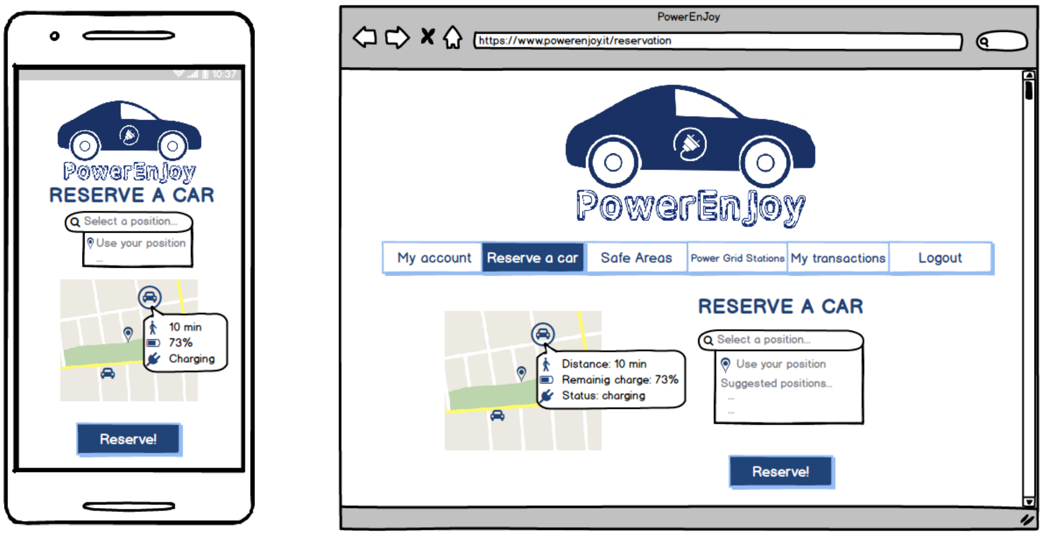


Figure 4: Reservation Page mockup

Once the USER has made his reservation, the page and application will show him a 1 hour timer, as shown in Figure 5, in order to remind him how much time he has left to go and pick up the reserved car. In this time, the USER can decide to cancel his prenotation by clicking on the button "*Cancel*". Only when the USER is nearby the car, he will be able to click the "*I'm there*" button to unlock the reserved car.

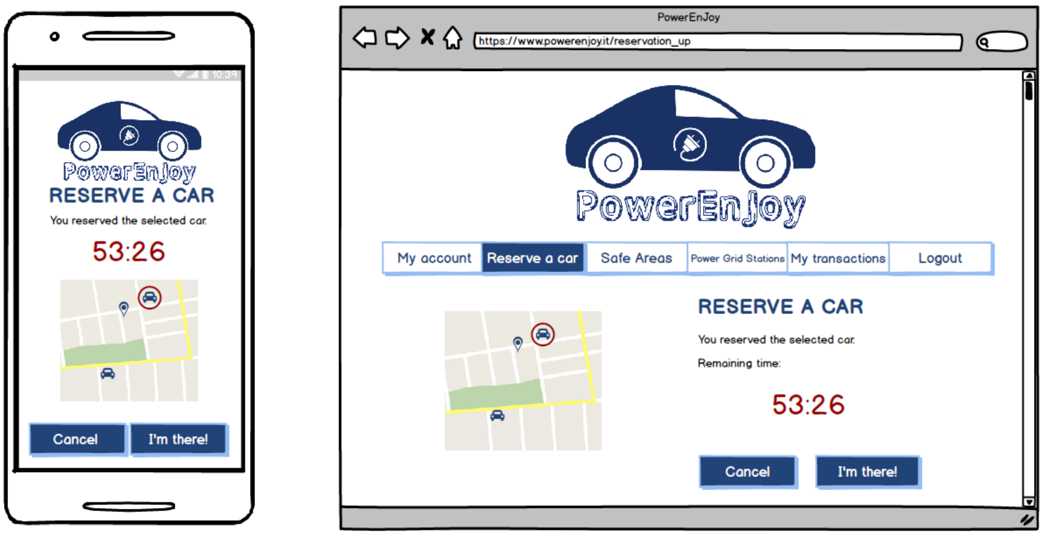


Figure 5: Reservation Timer Page mockup

If the car is not picked up within an hour, the reservation expires: the USER is reminded that he has to pay a fee of 1 EUR, as shown in Figure 6.

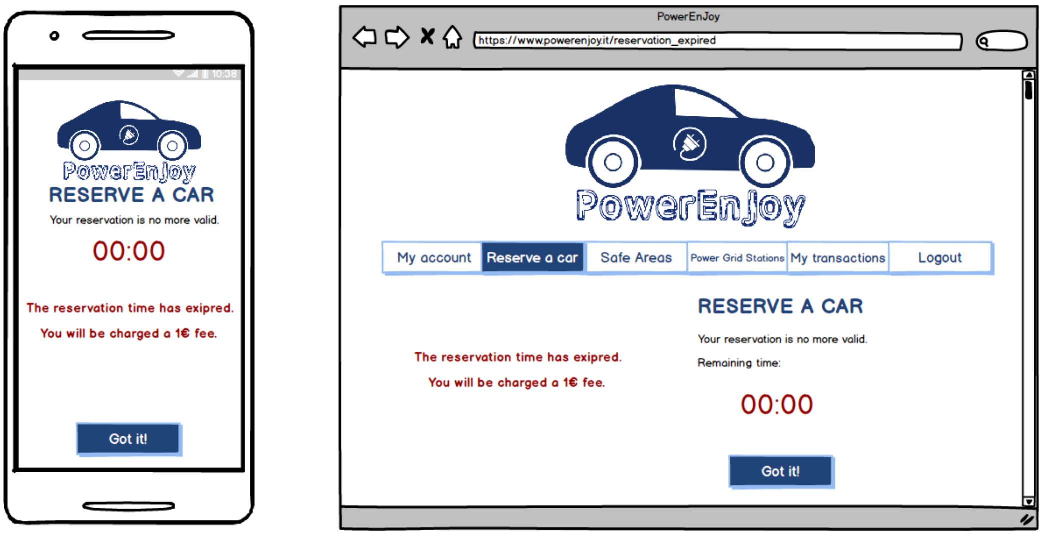

PowerEnJoy
RESERVE A CAR
Your reservation is no more valid.
00:00
The reservation time has expired.
You will be charged a 1€ fee.
Got it!
PowerEnJoy
https://www.powerenjoy.it/reservation_expired
PowerEnJoy
My account
Reserve a car
Safe Areas
Power Grid Stations
My transactions
Logout
RESERVE A CAR
Your reservation is no more valid.
Remaining time:
00:00
The reservation time has expired.
You will be charged a 1€ fee.
Got it!

Figure 6: Reservation Expired Page mockup

Safe Areas page

The mockup in Figure 7 shows the page where the USER can visualize the safe areas around him or around a selected adderess.

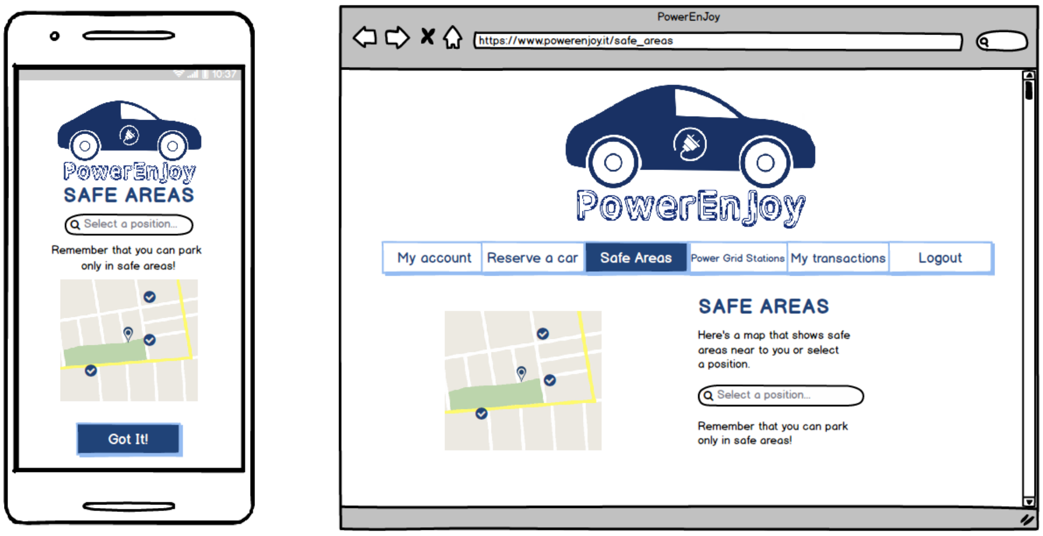
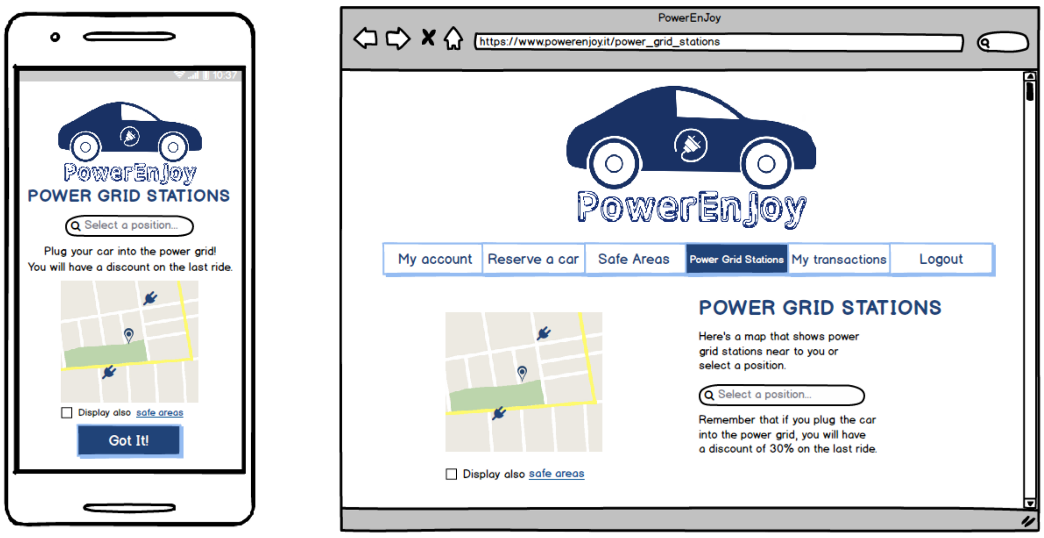

PowerEnJoy
SAFE AREAS
Select a position...
Remember that you can park only in safe areas!
Got It!
PowerEnJoy
https://www.powerenjoy.it/safe_areas
PowerEnJoy
My account
Reserve a car
Safe Areas
Power Grid Stations
My transactions
Logout
SAFE AREAS
Here's a map that shows safe areas near to you or select a position.
Select a position...
Remember that you can park only in safe areas!

Figure 7: Safe Areas Page mockup

Power Grid Stations page

The mockup in Figure 8 shows the page where the USER can visualize the power grid stations around him or around a selected adderess.



Plug your car into the power grid!
You will have a discount on the last ride.

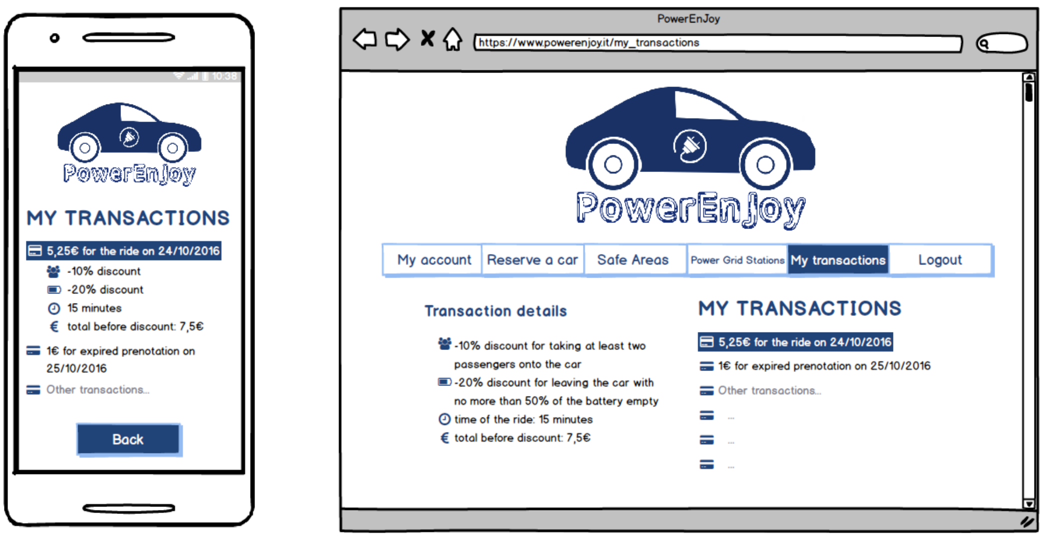
Here's a map that shows power grid stations near to you or select a position.

Remember that if you plug the car into the power grid, you will have a discount of 30% on the last ride.

Figure 8: Power Grid Station Page mockup

User's transactions page

The mockup in Figure 9 shows the page where the USER can visualize his transactions and the relative details, such as the date of the ride, its duration and the discount or the increase on the fee to be payed.



5,25€ for the ride on 24/10/2016
-10% discount
-20% discount
15 minutes
total before discount: 7,5€
1€ for expired prenotation on 25/10/2016
Other transactions...

Transaction details
-10% discount for taking at least two passengers onto the car
-20% discount for leaving the car with no more than 50% of the battery empty
time of the ride: 15 minutes
total before discount: 7,5€

5,25€ for the ride on 24/10/2016
1€ for expired prenotation on 25/10/2016
Other transactions...

Figure 9: User's Transactions Page mockup

Car Screen

The mockup in Figure 10 shows the car screen where useful information are visualized, for example the current level of charge, the duration of the rent and how much the USER has to pay before any discount or increase on the fee are applied.

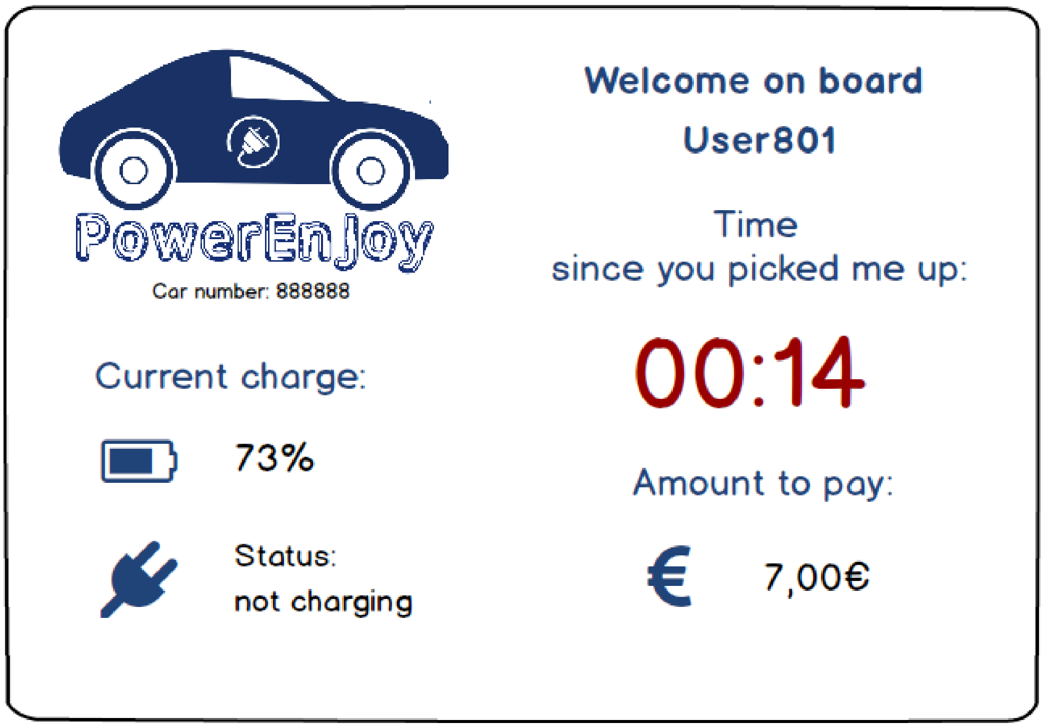


Figure 10: Car Screen mockup

3.1.2 Hardware Interfaces

Device should be enabled with Internet and GPS receiver.

3.1.3 Software Interfaces

The user's browser should be HTML5 compatible and the resolution should be at least 1280x720 for a satisfactory user experience.

3.2 Functional Requirements

In this section we explain all the functional requirements necessary to reach a specific goal. For each goal that was mentioned in the first section here we write its requirements.

G1 :

- –R1– During the registration, visitors can't choose an username already in use.
- –R2– During the registration, visitors can't choose an e-mail already in use.
- –R3– During the registration, visitors must insert valid credit card's information.
- –R4– During the registration, visitors must insert a valid driving license.
- –R5– During the registration, visitors must agree with the privacy conditions to use the system.

G2 :

- –R1– The registration must be successfully completed.
- –R2– The user must insert the correct old password.
- –R3– The user must repeat twice the new password.
- –R4– The user must upload a valid picture.

G3 :

- –R1– The user must be registered at the system.
- –R2– In the log in form the user must insert the correct username.
- –R3– In the log in form the user must insert the correct password.
- –R4– If the visitor insert the wrong data, the system shows an error and return to the log in page.

–R5– If the user doesn't remember the password he can receive it trough a special command.

G4 :

–R1– The user must insert a valid address or give to the system his position.

–R2– All cars position must be knowed.

–R3– All cars are set as available or not.

–R4– The list of nearest cars is made trough calculate the distance between car and user's position.

G5 :

–R1– The user must be correctly logged in.

–R2– The user must select which car he wants.

G6 :

–R1– The position of the user and the car must be the same.

–R2– The reservation must not be expired.

–R3– The system must recognize the correct car reserved by a specific user to unlock the correct one.

G7 :

–R1– The car's engine must ignites.

–R2– An amount of money per minute is set.

–R3– The final charge is based on the duration of the car use.

–R4– The system must shows all user's transactions.

–R5– The user must select a specific transaction to see its details.

G8 :

–R1– There must be a display on the car that communicates with the system.

–R2– The charge is update every minute.

G9 :

–R1– It must be passed one hour from the registration.

–R2– Is applyed a fee of 1 euro to the user involved.

–R3– The reservation is deleted.

–R4– In the page there is a timer.

G10 :

–R1– The user must be correctly logged in.

–R2– The user must go to the reservation page.

–R3– Mustn't be passed one hour from the reservation.

G11 :

–R1– The reservation is correctly deleted by a user or it is expired.

–R2– The car is set available in the list.

G12 :

–R1– A list of safe areas must be available to the user.

–R2– The car must be left in a safe area.

–R3– The user and all any passengers must exit from the car.

G13 :

–R1– The car must be used by a user.

–R2– Discounts are in a list where is defined all details.

–R3– In the car there is a sensor which counts how many people there are in the car, if they are three or more a discount is applied.

–R4– In the car there is a sensor which controls how many battery is empty, if it is no more than 50 per cent the system applied a discount.

–R5– If the user takes care to link the car to recharge the car he will have a discount.

G14 :

–R1– The car must have been recently used.

–R2– Most of 80 per cent of the battery is empty.

–R3– The car is left far away from a safe area.

G15 :

–R1– The user must go to the specific page.

–R2– The user must insert a correct position.

G16 :

–R1– The user must go to the specific page.

–R2– The user must insert a correct position.

3.3 Scenario Identification

3.3.1 Scenario 1: Registration and Log in

Monica is an ecologist girl who everyday goes at work by train because her work place is attainable only by train or by car. One day she goes to the train station but there is a strike, she is desperate: how can she go to work? Talking with other people in the station she learns that exists a car-sharing that provides only electric cars and that she can reserve a car trough an application on her mobile phone, this application is "*PowerEnJoy*". She quickly goes to the web site and download the app but to use it she must register giving some data, such as her driving license number and her credit card information. She inserts all the correct data and, as soon as she presses the registration button, in her mail box arrives a new e-mail with a password to access the system: she is now a new user. In the log in page she inserts her username and the password received so she can easily enter in the system. Monica is now in her personal page where she can change the password or insert a picture to personalize her profile. The first time you enter your personal page, you must change your password given you by the system with another one more secure and more easy to remember for the user.

3.3.2 Scenario 2: Reserve a car

Mario wants to reserve an electric car to go with his girlfriend to have a pic-nic in a beautiful meadow in mountain. He quickly enters in the "*PowerEnJoy*" app and goes to the reservation page that is very simple to use: he only must insert a position and the system will search for an available car nearby. In the box, he can insert his actual position activating his Smartphone GPS or insert a different one: he chooses the first option. In the map on the reservation page, it appears immediately that two cars are available. When he select one of them, he can see a curtain with the time distance between him and the car, the car's level of the battery and its status like "charging". He chooses the best one and pressed the button "***Reserve!***": immediately the reservation is confirmed and it appears a timer to remind him that he has one hour to go and pick up the car.

3.3.3 Scenario 3: Pick up a car

Luca has reserved a car and now, thanks to the map in the reservation page, he has walked towards the car. As soon as he is nearby the car, he presses the button "***I'm there!***". The system compares his position and the car's position: they are the same, so the car is unlocked and Luca can get into the car. There is a screen where he can see useful information: his username, the car's number, the battery level (so that he can know when he must charge the car) and, last but not least, the time passed from the moment he has picked up the car, with the corresponding amount to pay.

3.3.4 Scenario 4: Charge, park and lock a car

Since a long time, Maria has been using a car picked up through the "*PowerEnJoy*" app: she goes around the city for shopping, so she consume a lot of battery. She notices that the battery level is low and she must charge it. She knows that in the app there is a page where to consult the list of power grid station near a specific position: no one will remain with his car completly out of power. She accesses the app using her Smartphone and goes to the needed page. Then, she activates her GPS and in the map a power grid station appears: she can reach it quickly. When she arrives, she plug the car to re-charge it. After that, she continues her shopping. When she has finished all the commissions, she wants to park the car in a safe area so, just as she did before, she goes to the safe area's page, she chooses to use her position and a list of areas will appear on the map. She chooses for the best one and she goes there. As soon as she parks the car and exits from it, the system automatically locks the car.

3.3.5 Scenario 5: Transaction

Fabio is in a shop and, at the moment to pay, the shop assistant tell to him that he has reached the daily credit card's limit. He is furious and he cannot understand how it is possible. However, he suddenly reminds that in the morning he has picked up a car from "*PowerEnJoy*" and he thinks that the system has charged him more than necessary. Fortunately, in the app there is a page with the details of all the transaction. He logs in and goes to the transaction's page: there is the transactions list and, when one of the entry is selected, all its details appear. He selected the transaction corresponding to the morning reservation: the charge is correct because is specified the time of car using and the amount calculates from a specified cost per minute. However, he parked the car with only the 16 per cent battery remaing, so the sistem charged him with an additional 30 per cent of the cost of the last ride.

3.3.6 Scenario 6: Reservation expired

Giusy mobile phone starts ringing: it's her best friend who tells that she does not need anymore that Giusy go and pick her up to go to the party since she has already found an alternative solution. Giusy has already made a car reservation but she knows that she can delete it without problems because in the app reservation pageshe has the possibility to do this action. "There is no hurry" Giusy thinks, so she continues to works. She forgot that the reservation will last only one hour. The day after, she reminds she has not deleted the car reservation yet, so she takes her computer and goes to the site. But, when she arrives at the reservation's page, there is a surprise: the timer strikes 00:00 and there is a red written saying "*The reservation time is expired. You will be charged 1 euro fee.*". Next time she will pay more attention.

3.4 UML Models

3.4.1 Use Case Diagram

The Figure 11 shows the actors involved in the system, how they relate with it and all the ways in which they can use the functionalities provided by "*PowerEnJoy*".

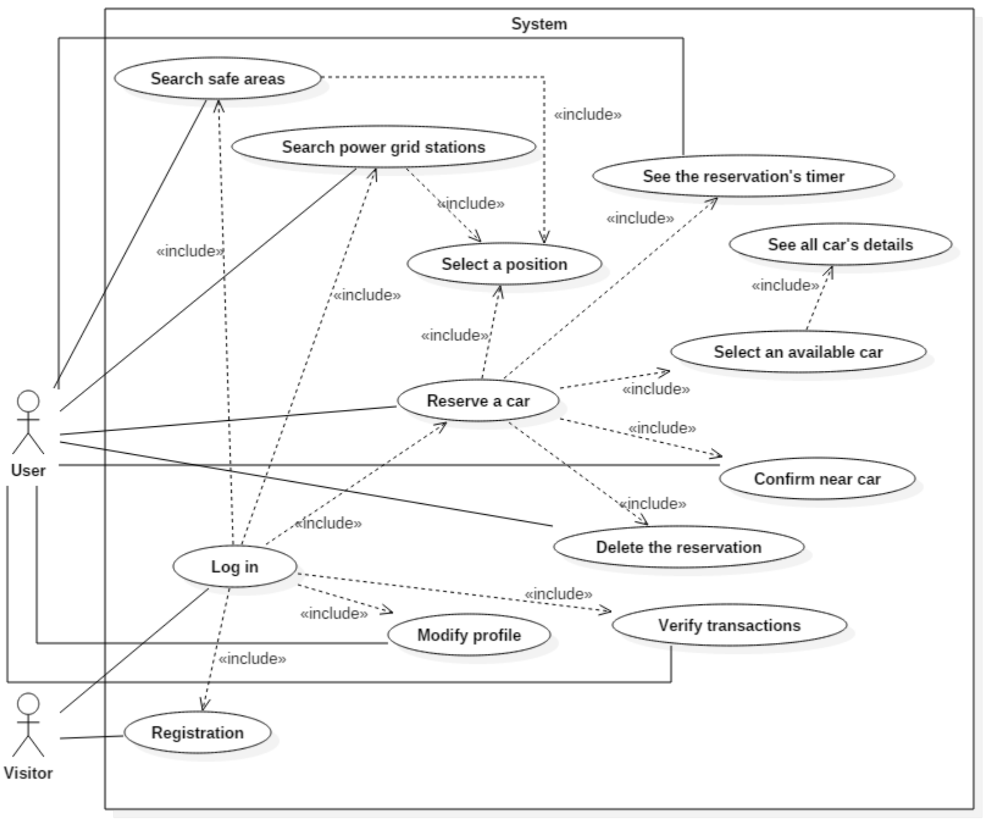


Figure 11: Use Case Diagram

3.4.2 Use Case Description

Registration

Actor	Visitor
Goal	**G1**
Requirements Associated	G1.**R1**,G1.**R2**,G1.**R3**,G1.**R4**,G1.**R5**
Precondition	The Visitor reach the registration form after access the application home page
Trigger	The Visitor has pressed the "Register Now!" button
Successful end condition	The Visitor is now a new User
Failed end condition	The Visitor is requested to try again because an error occurs
Main flow	1. The Visitor completes the registration form with all the necessary data. 2. The Visitor submits the form. 3. The system verifies that nickname and e-mail are not in used. 4. The system saves all the visitor's data in the database. 5. The system shows a confirmation message.
Extensions	
Inclusions	

Table 1: Registration Use Case

Log In

Actor	Visitor
Goal	**G3**
Requirements Associated	G3.**R1**,G3.**R2**,G3.**R3**,G3.**R4**,G3.**R5**
Precondition	The Visitor reach the log in form after correctly register at the system
Trigger	The Visitor has pressed the "Sign In!" button
Successful end condition	The Visitor is now an User and can do all the operations
Failed end condition	The Visitor doesn't insert the correct data or is not registered
Main flow	1. The Visitor inserts username and password. 2. The Visitor submits the log in form. 3. The system verifies the data. 4. The system shows the User page.
Extensions	
Inclusions	"Registration","Modify profile", "Verify transactions", "Reserve a car", "Search power grid stations", "Search safe areas"

Table 2: Log in Use Case

Modify profile

Actor	User
Goal	**G2**
Requirements Associated	G2.**R1**,G2.**R2**,G2.**R3**,G2.**R4**
Precondition	The User must be logged in
Trigger	The User has pressed the "Update" button
Successful end condition	The User's data are correctly upadated
Failed end condition	The new data are not correctly updated
Main flow	1. The Visitor inserts old password and/or a new picture. 2. The Visitor submits the update form. 3. The system verifies the data. 4. The system must save the new update in the database. 5. The system shows a confirmation message.
Extensions	
Inclusions	

Table 3: Modify profile Use Case

Verify transactions

Actor	User
Goal	**G7**
Requirements Associated	G7.**R4**,G7.**R5**
Precondition	The User must be logged in
Trigger	The User has selected a specific transaction from a list
Successful end condition	The User see all the selected transaction's details
Failed end condition	The User receives an error message
Main flow	1. The system shows all the user's transactions. 2. The User selects one form the list to see details.
Extensions	
Inclusions	

Table 4: Verify transaction Use Case

Search safe area

Actor	User
Goal	**G15**
Requirements Associated	G15.**R1**,G15.**R2**
Precondition	The User must be logged in
Trigger	The User has inserted a correct position.
Successful end condition	The User see all the nearest safe areas
Failed end condition	The User receives an error message
Main flow	1. The user inserts a correct position or activates the GPS. 2. The system shows on the map the safe areas. 3. The system must have a list of all the safe areas.
Extensions	
Inclusions	"Select a position"

Table 5: Search safe area Use Case

Search power grid station

Actor	User
Goal	**G16**
Requirements Associated	G16.**R1**,G16.**R2**
Precondition	The User must be logged in
Trigger	The User has inserted a correct position.
Successful end condition	The User see all the nearest power grid stations
Failed end condition	The User receives an error message
Main flow	1. The user inserts a correct position or activate the GPS. 2. The system shows on the map the power grid stations. 3. The system must have a list of all the power grid stations.
Extensions	
Inclusions	"Select a position"

Table 6: Search power grid station Use Case

Reserve a car

Actor	User
Goal	**G4**,**G5**
Requirements Associated	G4.**R1**,G4.**R2**,G4.**R3**,G4.**R4**, G5.**R1**,G5.**R2**
Precondition	The User must be logged in
Trigger	The User has pressed the "Reserve!" button
Successful end condition	The User see all the nearest available cars
Failed end condition	The User receives an error message
Main flow	1. The user inserts a correct position or activate the GPS. 2. The system shows on the map the available car. 3. The User must select one of the cars. 4. The system must shows all the selected car's detail. 5. The User must submit the reservation form. 6. The system saves all the reservation's details and start shows a timer.
Extensions	
Inclusions	"Select a position","Delete the reservation", "Confirm near car", "Select an available car"(that includes "See all car's details"), "See the reservation's timer"

Table 7: Reserve a car Use Case

Delete a reservation

Actor	User
Goal	**G10**,**G11**
Requirements Associated	G10.**R1**,G10.**R2**,G10.**R3**,G11.**R1**,G11.**R2**
Precondition	The User must have made a reservation
Trigger	The User has pressed the "Delete!" button
Successful end condition	The User receives a confirmation message
Failed end condition	The User receives an error message
Main flow	1. The user go to the reservation page. 2. The reservation must not be expired. 3. The system must set the car as available.
Extensions	
Inclusions	

Table 8: Delete a reservation Use Case

Confirm to be near the reserved car

Actor	User
Goal	**G6**
Requirements Associated	G6.**R1**,G6.**R2**,G6.**R3**
Precondition	The User must have made a reservation and must be near the car
Trigger	The User has pressed the "I'm There!" button
Successful end condition	The car is unlocked
Failed end condition	The car remains locked
Main flow	1. The user goes to the reservation page. 2. The reservation must not be expired. 3. The position of the user and the car must be the same.
Extensions	
Inclusions	

Table 9: Confirm to be near the reserved car Use Case

See the reservation timer

Actor	User
Goal	**G9**
Requirements Associated	G9.**R4**
Precondition	The User must have made a reservation
Trigger	The User goes to the reservation page
Successful end condition	The timer is showed
Failed end condition	The User receives an error message
Main flow	1. The user goes to the reservation page. 2. The user must have made a reservation.
Extensions	
Inclusions	

Table 10: See the reservation timer Use Case

3.4.3 Class Diagram

In Figure 12 is shown the Class diagram, a static diagram that describes our system structure by showing classes, attributes and their relationships. The methods will be inserted during the developing process.

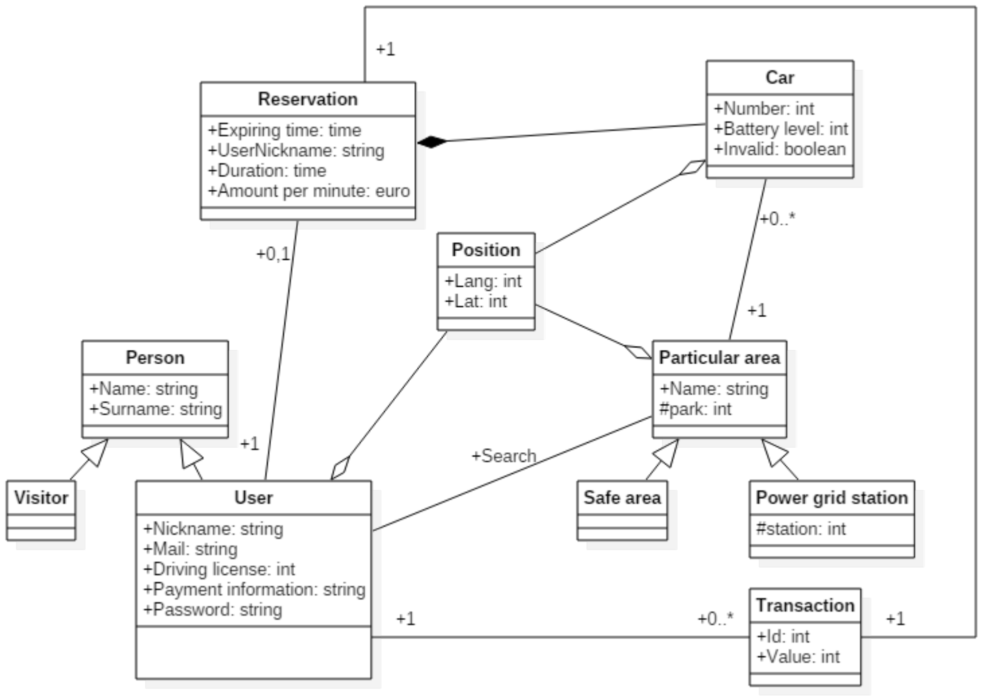


Figure 12: Class Diagram

3.4.4 Sequence Diagram

Here we rappresent some Sequence diagrams usefull to understand the dynamic evolution of the system.

Reserve a car

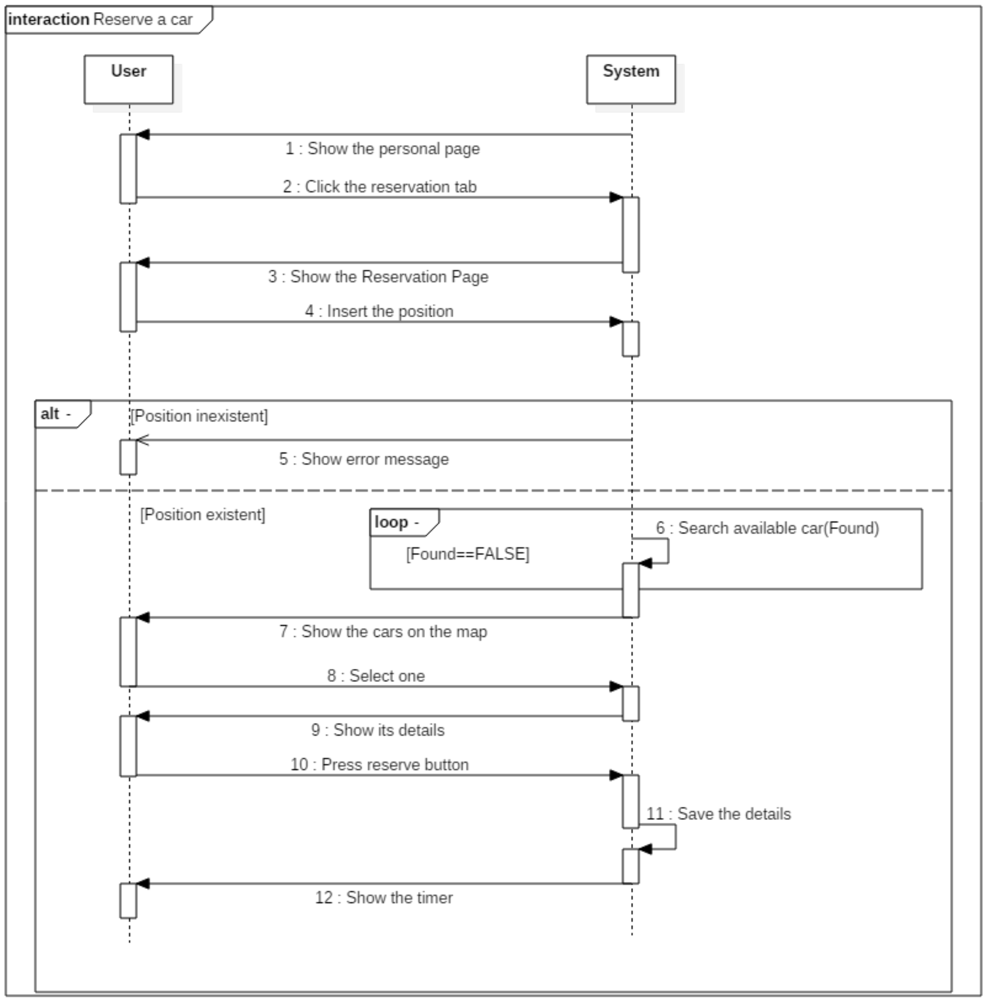


Figure 13: Reserve Sequence Diagram

Near the reserved car

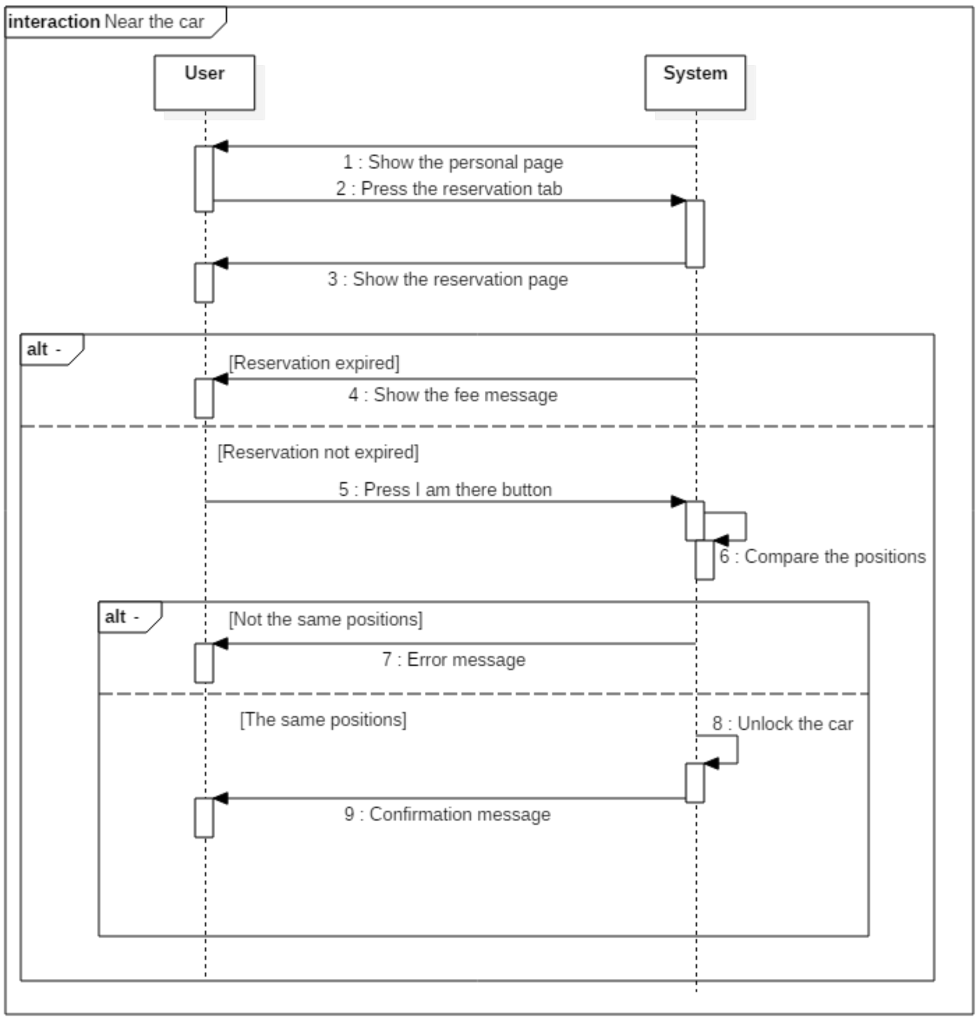
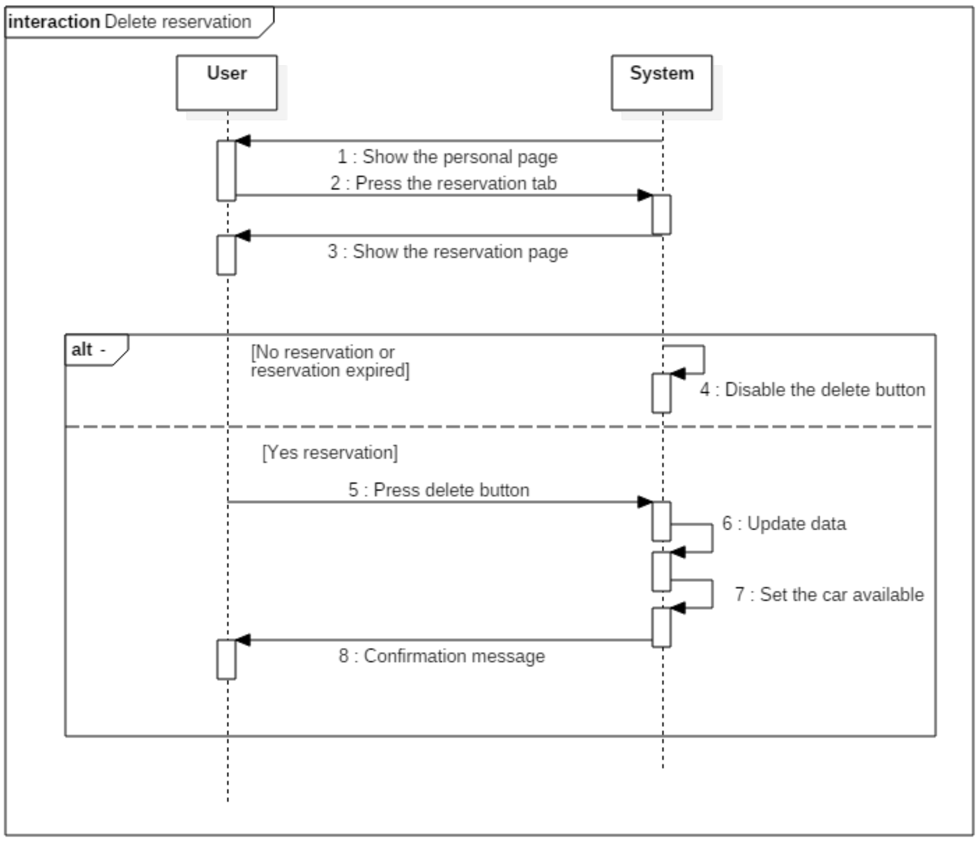


Figure 14: Near the car Sequence Diagram

Delete a reservation



interaction Delete reservation
User
System
1 : Show the personal page
2 : Press the reservation tab
3 : Show the reservation page
alt -
[No reservation or reservation expired]
4 : Disable the delete button
[Yes reservation]
5 : Press delete button
6 : Update data
7 : Set the car available
8 : Confirmation message

Figure 15: Delete Reservation Sequence Diagram

See transaction details

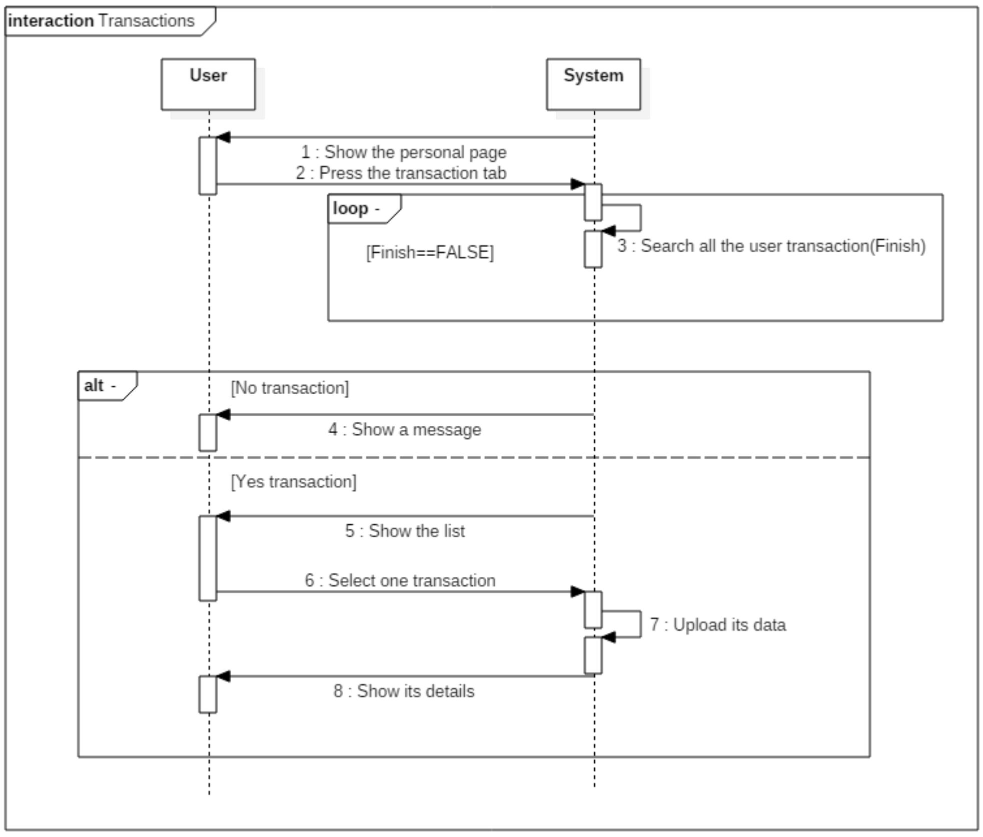


Figure 16: Transactions Sequence Diagram

3.4.5 State Chart Diagram

User state chart diagram

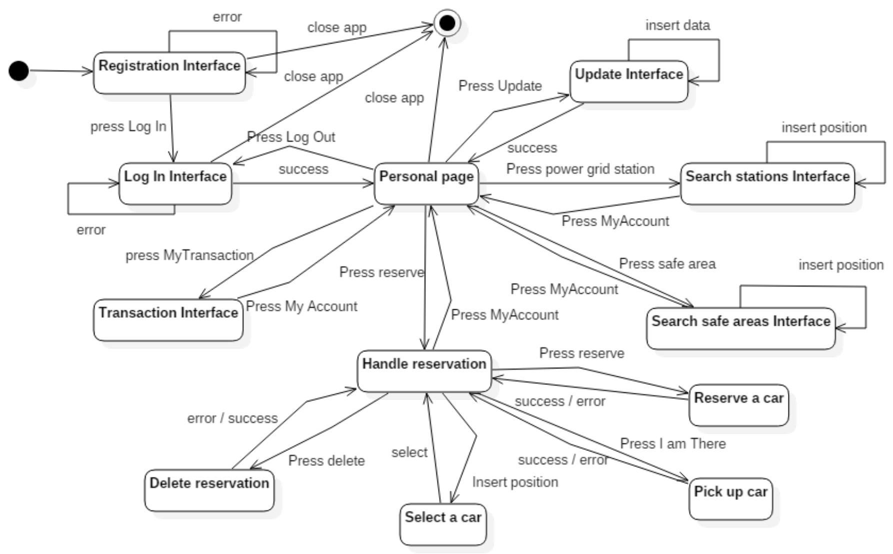


Figure 17: User State Chart Diagram

3.5 Performance Requirements

Performance must be acceptable to garantee a quiet good level of usability, in fact we supposed the time between the reservation confirmation and the timer start is equal to zero as well as the time between the user arrival near the car and its unlock. The system performance is closely linked with user internet connection and so indipendent from our develop.

3.6 Design Constraints

All design constraints are linked with the language used to program.

3.7 Software System Attributes

3.7.1 Availability

Our system can be accessed in each time the user wants, it's only needs the internet connection so its availability is closely related to it presence. The fact that the system is always accessible does not guarantee the presence of available car: this is indipendent from our system.

3.7.2 Security

The user's data are all protected by the use of password and database's cryptography trough MD5 function. The transaction must be authorized by the user's bank and all its details are consultable in a user's page called "*My Transaction*" so the user can be sure that his credit card will not be used illegally.

3.7.3 Maintainability

The system maintainability depends on the API's develop but thanks to all the document it will be simple in the future to adapt the software to any change.

3.7.4 Portability

The application can be used to any OS so it has a very good portability.

4 Alloy Modeling

4.1 Alloy Code

```
/*******************  CLASSES  *******************/

// power grid station status
abstract sig StationStatus {}
// the power grid station is free
one sig Free extends StationStatus {}
// there is a car plugged into the power grid station
one sig InUse extends StationStatus {}

// battery level of the car
abstract sig BatteryLevel {}
// the level of the battery is high (>50%)
one sig High extends BatteryLevel {}
// the level of the battery is medium
one sig Medium extends BatteryLevel {}
// the level of the battery is low (<20%)
one sig Low extends BatteryLevel {}

// indicate if the car is plugged into a power grid station
abstract sig ChargingStatus {}
// the car is plugged into a power grid station
one sig Charging extends ChargingStatus {}
// the car is not plugged into a power grid station
one sig NotCharging extends ChargingStatus {}

// status of the car
abstract sig CarStatus {}
// the car is parked and somone reserved it
one sig Reserved extends CarStatus {}
// the car is parked and none reserved it
one sig NotReserved extends CarStatus {}
// the user picked up the car
one sig BeingDrived extends CarStatus {}

// distance of the car from the nearest power grid station
abstract sig Distance {}
// the car is far (distance >3Km)
one sig Far extends Distance {}
// the car is near (distance <= 3Km)
one sig Near extends Distance {}

// status of the reservation
abstract sig ReservationStatus {}
// the user has one hour to go and pick up the car
one sig Valid extends ReservationStatus {}
// the user picked up the car
one sig UserDriving extends ReservationStatus {}
// the user parked the car after the ride
one sig EndOfTheRide extends ReservationStatus {}
// the user let the resevation expire
one sig Expired extends ReservationStatus {}

sig Position {}

sig User {}
```

```
// parking area for a certain number of car
// (in this model, for simplicity, we consider only one car)
sig SafeArea {
        position: one Position
}

// the parking area can have  a certain number of power grid station where a
    ↪  car can be plugged in
// (in this model, for simplicity, we consider only one power grid statio)
sig PowerGridStation extends SafeArea {
        status: one StationStatus
}

abstract sig Car {
        status: one CarStatus,
        position: one Position,
        batteryLevel: one BatteryLevel,
}

// a car that someone is driving
sig UnlockedCar extends Car {}

// a car that can be reserved or not, it can be connected to a power grid
    ↪ station (charging) or it can be at some distance
// from a power grid station (distance)
sig LockedCar extends Car {
        chargingStatus: one ChargingStatus,
        distance: one Distance
}

sig Passenger {}

sig Reservation {
        user: one User,
        reservedCar: one Car,
        reservationStatus: one ReservationStatus,
        passengers: some Passenger
}

sig Transaction {
        owner: one User,
        reservation: one Reservation,
        fee: Int
}

/******************  CONSTARAINTS  ******************/

// 1 - ensure that all locked car are in a safe area
fact AllLockedCarInASafeArea{
        all c: LockedCar | one s: SafeArea | c.position = s.position
}

// 2 - ensure that all locked car marked as charging are in a safe area with
    ↪  a power grid station
fact AllChargingCarInASafeAreaWithPowerGridStation{
        all c: LockedCar | one p: PowerGridStation | c.chargingStatus =
            ↪ Charging implies c.position = p.position
}

// 3 - ensure that all unlocked car are not in a safe area
fact AllUnlockedCarNotInASafeArea{
        all c: UnlockedCar | all s: SafeArea | c.position ≠ s.position
```

```
}

// 4 - ensure that a locked car is not in the same safe area of another car
fact OnlyOneLockedCarPerSafeArea{
        all disj c1, c2: LockedCar | c1.position ≠ c2.position
}

// 5 - ensure that in all power grid station marked as not free there is a
    ↪ car charging
fact AllPowerGridStatioNotFree{
        all p: PowerGridStation | one c: LockedCar | c.position = p.position
            ↪  ∧ (c.chargingStatus = Charging iff p.status = InUse)
}

// 6 - ensure that a locked car in the same position of a power grid station
    ↪  is flagged as "Near"
fact AllLockedCarInASafeAreaWithPowerGridStationFlaggedAsNear{
        all c: LockedCar | one p: PowerGridStation | c.position = p.position
            ↪  implies c.distance = Near
}

// 7 - ensure that there is only one safe area in one position
fact OnlyOneSafeAreaInOnePosition{
        all disj s1, s2: SafeArea | s1.position ≠ s2.position
}

// 8 - ensure that there is only one unlocked car in one position
fact OnlyOneSafeAreaInOnePosition{
        all disj u1, u2: UnlockedCar | u1.position ≠ u2.position
}

// 9 - ensure that all unlocked cars are associated to a reservation
fact AllUnlockedCarAreReserved{
        all u: UnlockedCar | one r: Reservation | r.reservedCar = u
}

// 10 - ensure that all locked cars flagged as "Reserved" are associated to
    ↪ a reservation
fact AllReservedLockedCarAreAssociatedToAReservation{
        all c: LockedCar | one r: Reservation | c.status = Reserved implies
            ↪ r.reservedCar = c
}

// 11 - ensure that all distinct reservation refers to different cars
fact AllReservationAreForDifferentCars{
        all disj r1, r2: Reservation | r1.reservedCar ≠ r2.reservedCar
}

// 12 - ensure that all distinct reservation refers to different users
fact AllReservationAreForDifferentUsers{
        all disj r1, r2: Reservation | r1.user ≠ r2.user
}

// 13 - ensure that all distinct reservation refers to different passengers
fact AllReservationAreForDifferentPassengers{
        all disj r1, r2: Reservation | disj[r1.passengers , r2.passengers]
}

// 14 - ensure that all valid reservation are associated to a reserved
    ↪ locked car
fact AllValidReservationAreAssociatedToALockedCar{
        all r: Reservation | r.reservationStatus = Valid implies r.
            ↪ reservedCar.status = Reserved
}

// 15 - ensure that all reservation flagged as "Expired" or "EndOfTheRide"
    ↪ are associated to a not reserved locked car
fact AllValidReservationAreAssociatedToALockedCar{
        all r: Reservation | (r.reservationStatus = Expired or r.
            ↪ reservationStatus = EndOfTheRide) implies r.reservedCar.
            ↪ status = NotReserved
}

// 16 - ensure all unlocked car are flagged as "BeingDrived"
fact AllUnlockedCarAreFlaggedAsBeingDrived{
        all u: UnlockedCar | u.status = BeingDrived
}

// 16a - ensure all locked car are not flagged as "BeingDrived"
fact AllLockedCarAreNotFlaggedAsBeingDrived{
        all c: LockedCar | c.status ≠ BeingDrived
}

// 17 - ensure that all reservation flagged as "UserDriving" are associated
    ↪ to an unlocked car
fact AllValidReservationAreAssociatedToALockedCar{
        all r: Reservation | r.reservationStatus = UserDriving iff r.
            ↪ reservedCar.status = BeingDrived
}

// 18 - ensure that all transactions have the same owner of the
    ↪ corresponding reservation
fact AllTransactionAndCorrespondingReservationWithSameUser{
        all t: Transaction | t.owner = t.reservation.user
}

// 19 - ensure that all transactions have a non negative fee
fact AllTransactionsHaveANotNegativeFee{
        all t: Transaction | t.fee ≥ 0
}

// 20 - ensure that all transactions refers to different reservation
fact AllTransactionRefersToDifferentReservations{
        all disj t1, t2: Transaction | t1.reservation ≠ t2.reservation
}

// 21 - ensure that all reservation are related to a transaction
fact AllReservationRelatedToATransaction{
        all r: Reservation | one t: Transaction | t.reservation = r
}

// 22 - ensure that all transaction that refers to "Valid" reservation have
    ↪ fee equal to zero
// -> the ride is not started yet: the fee will be once the user starts
    ↪ driving
fact AllTransactionThatRefersToAValidOrUserDrivingReservationHaveFeeZero{
        all t: Transaction | (t.reservation.reservationStatus = Valid)
            ↪ implies t.fee = 0
}

// 23 - ensure that all transaction that refers to an "Expired" reservation
    ↪ have fee equal to one
fact AllTransactionThatRefersToAExpiredReservationHaveFeeOne{
        all t: Transaction | (t.reservation.reservationStatus = Expired)
            ↪ implies t.fee = 1
}

// 24 - ensure that all transaction that refers to a "UserDriving" or "
    ↪ EndOfTheRide" reservation have fee greater than one
fact AllTransactionThatRefersToAEndOfTheRideReservationHaveFeeGreaterThanOne
    ↪ {
        all t: Transaction | (t.reservation.reservationStatus = UserDriving
            ↪ or t.reservation.reservationStatus = EndOfTheRide) implies t.
            ↪ fee > 1
}

/******************  ASSERTIONS  ******************/

// 1 - check that all locked car marked as charging are in a safe area with
    ↪ a power grid station
assert AllChargingCarInASafeAreaWithPowerGridStation{
                        no c: LockedCar | all p: PowerGridStation | c.
                            ↪ chargingStatus = Charging ∧ c.position ≠ p.
                            ↪ position
}
check AllChargingCarInASafeAreaWithPowerGridStation

// 2 - check that a locked car is not in the same safe area of another car
assert OnlyOneLockedCarPerSafeArea{
                        no disj c1, c2 : LockedCar | c1.position = c2.
                            ↪ position
}
check OnlyOneLockedCarPerSafeArea

// 3 - check that there is only one safe area in one position
assert OnlyOneSafeAreaInOnePosition{
                        no disj s1, s2: SafeArea | s1.position = s2.position
}
check OnlyOneSafeAreaInOnePosition

// 4 - check that there is only one unlocked car in one position
assert OnlyOneUnlockedCarInOnePosition{
                        no disj u1, u2: UnlockedCar | u1.position = u2.
                            ↪ position
}
check OnlyOneUnlockedCarInOnePosition

// 5 - check that all distinct reservation refers to different cars
assert AllReservationAreForDifferentCars{
                        no disj r1, r2: Reservation | r1.reservedCar = r2.
                            ↪ reservedCar
}
check AllReservationAreForDifferentCars

// 6 - check that all distinct reservation refers to different users
assert AllReservationAreForDifferentUsers{
                        no disj r1, r2: Reservation | r1.user = r2.user
}
check AllReservationAreForDifferentUsers

// 7 - check that all transactions have the same owner of the corresponding
    ↪ reservation
assert AllTransactionAndCorrespondingReservationWithSameUser{
                        no t: Transaction | t.owner ≠ t.reservation.user
}
```

```
check AllTransactionAndCorrespondingReservationWithSameUser


// 8 - check that all transactions have a positive fee
assert AllTransactionsHaveAPositiveFee{
                        no t: Transaction | t.fee < 0
}
check AllTransactionsHaveAPositiveFee

// 9 - check that all transactions refers to different reservation
assert AllTransactionRefersToDifferentReservations{
                        all disj t1, t2: Transaction | t1.reservation ≠ t2.
                            ↪ reservation
}
check AllTransactionRefersToDifferentReservations

/*******************  PREDICATES  *******************/

pred showAll() {
                        #UnlockedCar > 1
}
run showAll for 5 but exactly 3 LockedCar

pred showLockedCar() {}
run showLockedCar for 5 but exactly 3 LockedCar

pred showUnlockedCar() {}
run showUnlockedCar for 3 but exactly 3 UnlockedCar
```

4.2 Worlds generated

The following images will show you the world generated from the execution of the predicate showAll(), ShowLockedCar() and ShowUnlockedCar(), respectively.

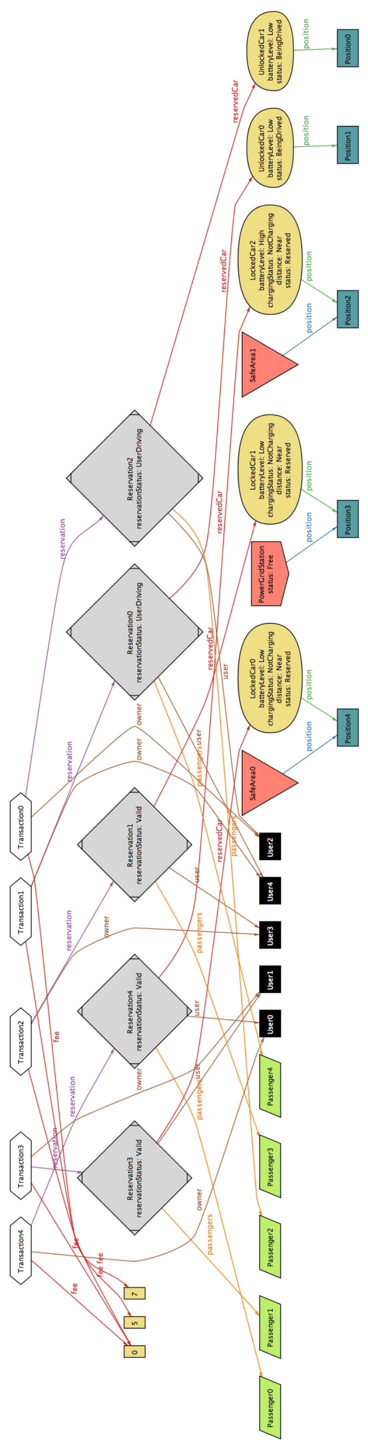

UnlockedCar1 batteryLevel: Low status: BeingDrived
UnlockedCar0 batteryLevel: Low status: BeingDrived
LockedCar2 batteryLevel: High chargingStatus: NotCharging distance: Near status: Reserved
SafeArea1
LockedCar1 batteryLevel: Low chargingStatus: NotCharging distance: Near status: Reserved
PowerGridStation status: Free
LockedCar0 batteryLevel: Low chargingStatus: NotCharging distance: Near status: Reserved
SafeArea0
Position0
Position1
Position2
Position3
Position4
Reservation2 reservationStatus: UserDriving
Reservation0 reservationStatus: UserDriving
Reservation1 reservationStatus: Valid
Reservation4 reservationStatus: Valid
Reservation3 reservationStatus: Valid
Transaction0
Transaction1
Transaction2
Transaction3
Transaction4
User2
User4
User3
User1
User0
Passenger4
Passenger3
Passenger2
Passenger1
Passenger0
7
5
0
reservedCar
position
reservation
owner
user
passengers
fee

Figure 18: World generated from the execution of predicate showAll()

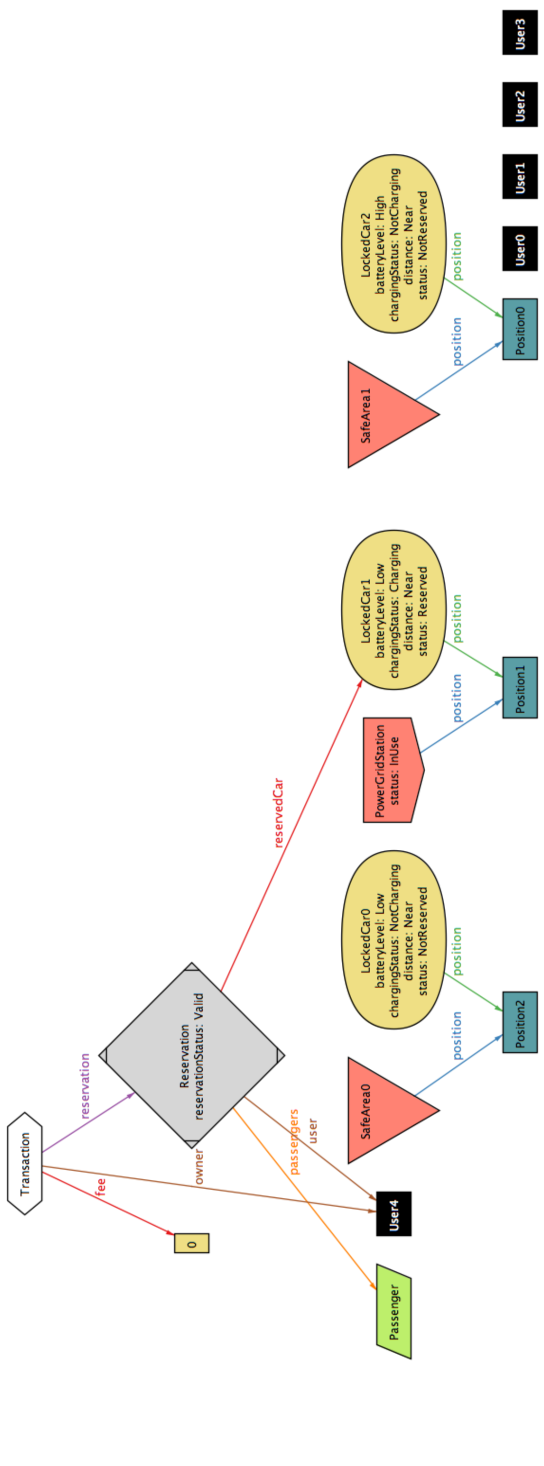


Figure 19: World generated from the execution of predicate ShowLockedCar()

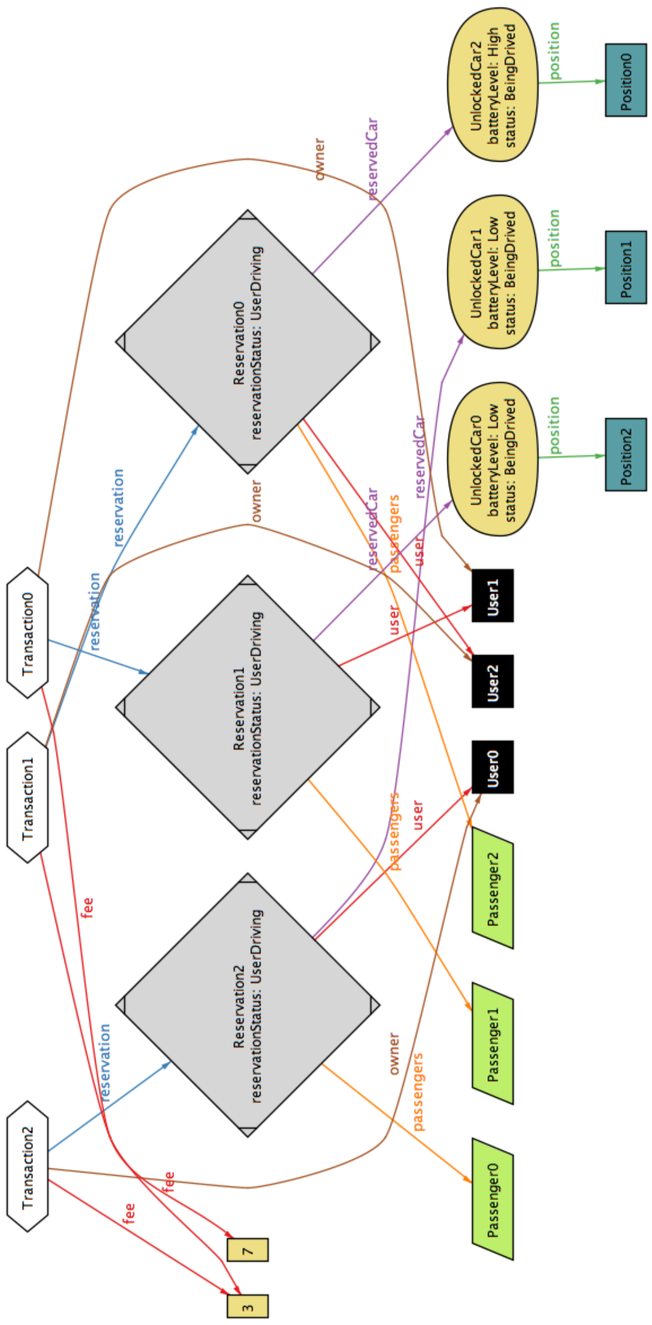


Figure 20: World generated from the execution of predicate ShowUnlocked-Car()

5 Appendix

In this section, we will give the information about the used tools, the hours of work done by the members of the group.

5.1 Used Tools

In this first phase of the project, the following tools have been used:

- LaTeX and TeXMaker editor: to redact and to format this document
- Balsamiq Mockups (`https://balsamiq.com`): to create the mockups
- StarUML (`http://staruml.io`): to create the State Charts, the Class Diagram, the Sequence Diagrams, the Use Case Diagram
- Alloy Analizer (`http://alloy.mit.edu/alloy`): to prove the consistency of our model

5.2 Working Hours

Last Name	**First Name**	**Total Hours**
Blanco	Federica	27 h
Casasopra	Fabiola	27 h

5.3 RASD Modifications

- Delete the requirement G5.R3: The user must select only date and times valid for the reservation.
- Add some specifications about the payment actions: Domain **D9** and assumption number 13.