Politecnico di Milano

Computer Science and Engineering's master degree course

PowerEnjoy
Requirements Analysis and Specification Document

Version 1.0

Release Date: 13/11/2016

Customer: Eng. Elisabetta DI NITTO

Authors:

Eng. Marco FERNI	Id. 877712
Eng. Angelo Claudio RE	Id. 877808
Eng. Gabriele TERMIGNONE	Id. 877645

Contents

Chapter 1

Introduction

1.1 Purpose

This document has the aim to show the functional and non-functional requirements of the system-to-be, based on several important aspects: the needs expressed by the stakeholders, the constraints which it is subject to, the typical scenarios that will happen after its deployment and the consistency's verification of the system-to-be. The intended audience is mainly made of software engineers and developers who have to actually develop the service here described.

1.2 Scope

We are to develop a digital management system for a car-sharing service that exclusively employs electric cars. First, the system should provide the functionality normally provided by car-sharing services. These include:

- Users must be able to register to the system by providing their credentials and payment information. 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.
- Among the available cars in a certain geographical region, users must be able to reserve a single car for up to one hour before they pick it up.
- If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; the user pays a fee of 1 EUR.
- A user that reaches a reserved car must be able to tell the system she's nearby, so the system unlocks the car and the user may enter.
- As soon as the engine ignites, the system starts charging the user for a given amount of money per minute; 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; at this point, the system locks the car automatically.
- The set of safe areas for parking cars is pre-defined by the management system.

- If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.
- If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.
- If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.
- If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.
- If the user enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

1.3 Definition, acronyms, abbreviations

"Cost of the trip" is the raw price of the service calculated only on the base of the duration of the car's usage, before discounts or additional charges are applied.

"Virtuousness coefficient" is the factor by which to multiply the cost of the trip to get the amount of the bill. Its initial value is 1.

"Supervisor" is a company employee who work at the Car hub controller.

"Recharge on site" is a company procedure: a worker is sent to recharge a low car that was parked detached from the power grid.

"Car recovery" is a company procedure: a worker is sent to retrieve a car that has been forgotten outside a safe area and park in one of these.

"Guest" is a person who is not already registered to the system.

"User" is a registered customer.

"RASD" is the acronym of Requirement Analysis and Specification Document

1.4 Reference documents

- Assignment 1 of *"Assignments AA 2016-2017.pdf"*

- *"IEEE standard on requirement engineering.pdf"*

- *"RASD SAMPLE FROM OCT. 20 LECTURE.pdf"*

1.5 Overview

The document is organized in 3 main chapter. The First chapter describes the main assumptions done, with regarding to the customer needing and likely constraints. The second chapter is focused on the description of the requirements, goals, mockups and uml diagrams of the system-to-be. The last chapter illustrates the revisions notes of current document, the tools used to lead this analysis and the effort spent of the authors of the analysis.

Chapter 2

Overall Description

2.1 Product perspective

2.1.1 Class Diagrams

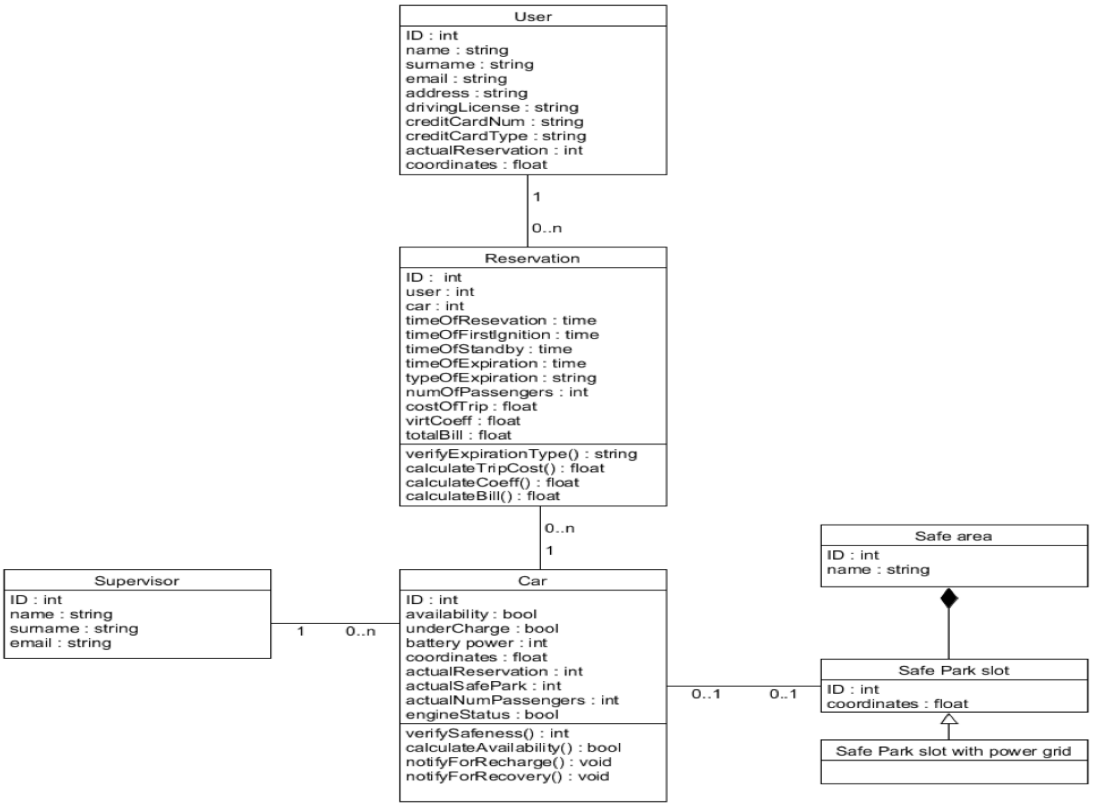

Figure 2.1: *PowerEnjoy Class Diagram*

2.1.2 Statecharts

The following UML diagram shows the control flow of an available car

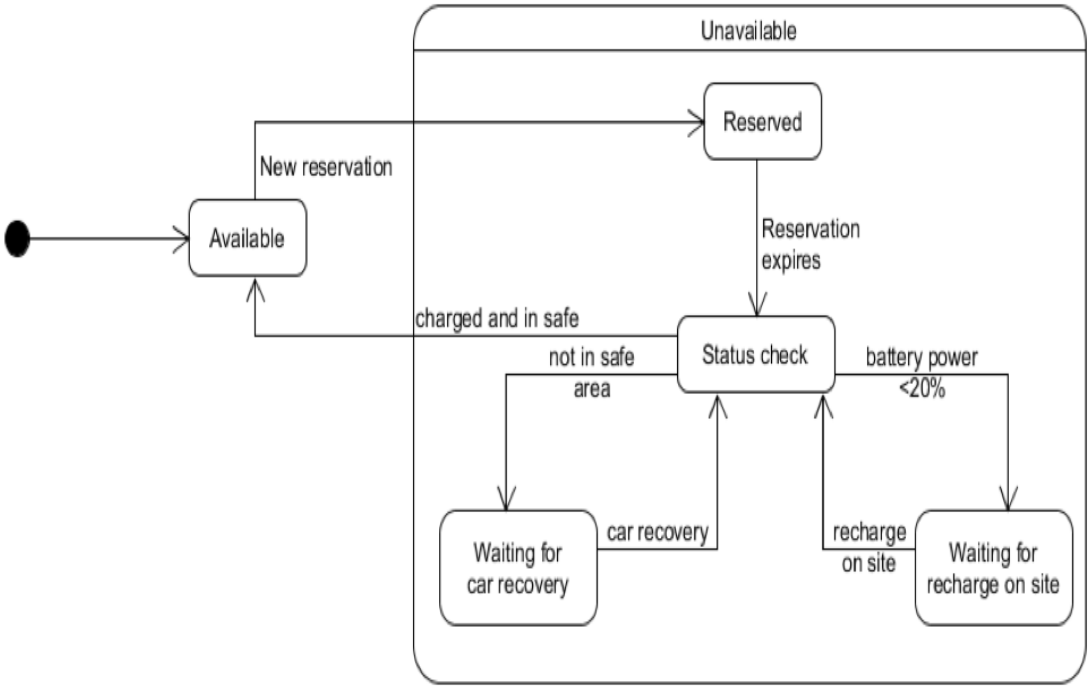

Figure 2.2: *Car's Availability Statechart*

The following UML diagram shows the control flow of the user reservation of an available car

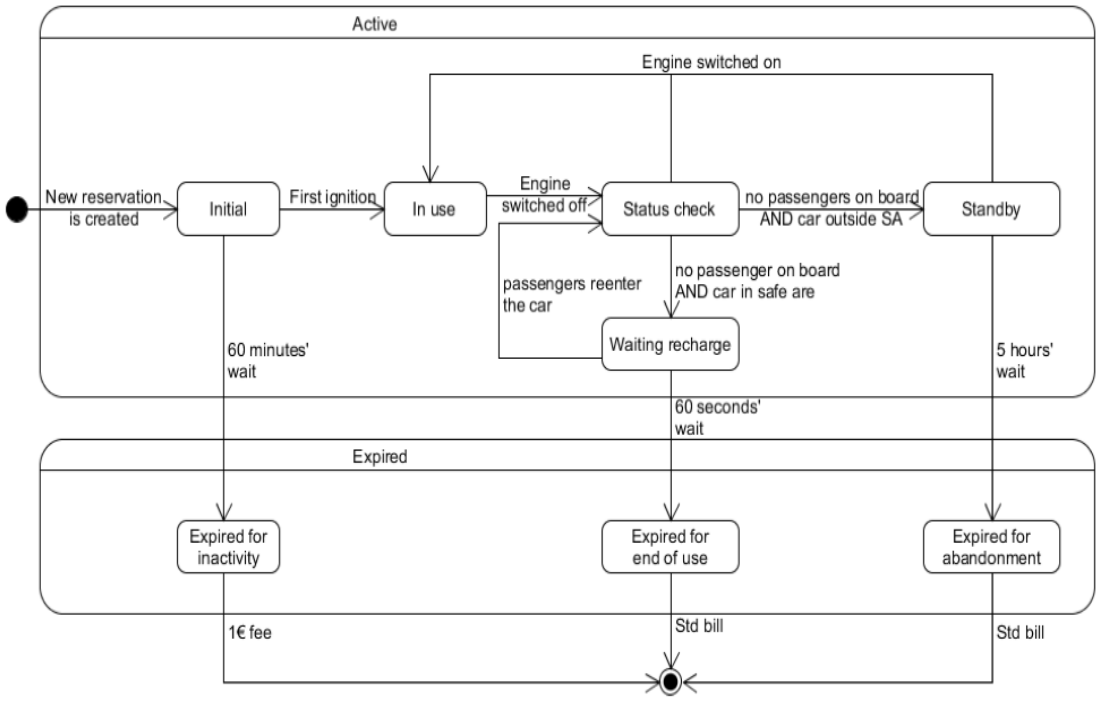

Figure 2.3: *Reservation Statechart*

2.2 Assumptions and Dependencies

It was stated the following domain assumptions:

- The only payment methods accepted are credit card (not prepaid).
- The system continues counting the minutes of usage while the car is outside the safe areas, also if it's turned off.
- A user uses a car only if the power remaining in the car's battery is sufficient for his trip.
- There is a CAR HUB CONTROLLER where a "supervisor" can monitor the status of every car and dispatch the "recharge on site" if a car is left with less than 20% battery life remaining and it isn't connected to a power grid. The supervisor is also responsible to dispatch the "car recovery responsible for" when a car is left outside a safe area for more than 5 hours.
- The cars can be recharged by the users only in the special parking areas with power grid.
- Special parking areas with power grid are a subset of the predefined set of the safe areas.
- When a user get into a car, he actually starts using it.
- A user finishes using the car when he leaves the car in a safe area.
- Users are not permitted to delete a reservation.
- When users get in or exit a car, they close the car's door.
- Users do not leave the car without turning it off.
- The passengers counted to obtain the 10% discount are them who are present in the car at the time of the first ignition.

- The screens on the car show the same infos showed on the "myReservation" page of the users' app (duration of the trip, current cost of the trip).

Chapter 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Here is the main mockups examples of likely *look and feel* that a user of the system could exploit:

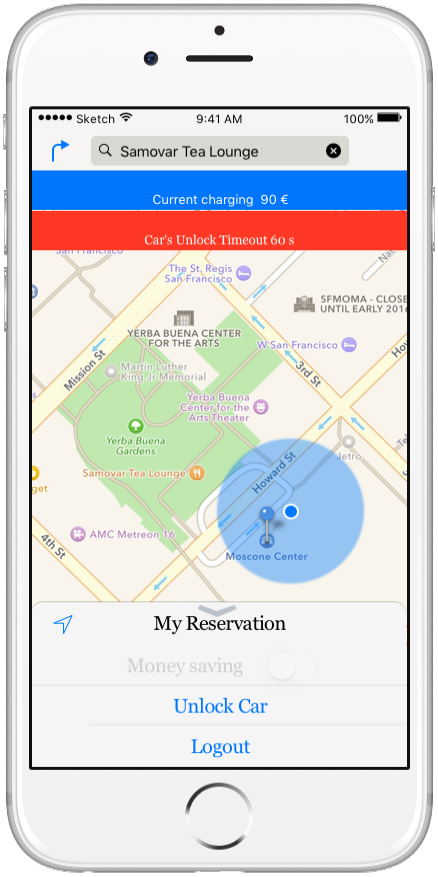

Figure 3.1: *car in the Ikea's parking lot*

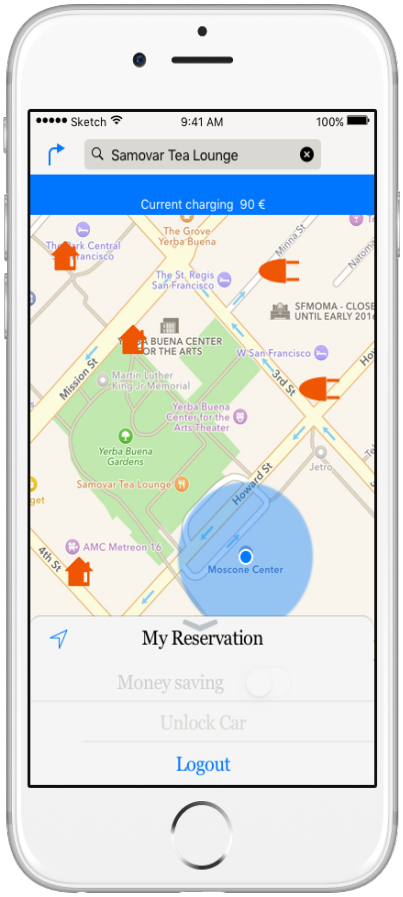

Figure 3.2: *car on move*

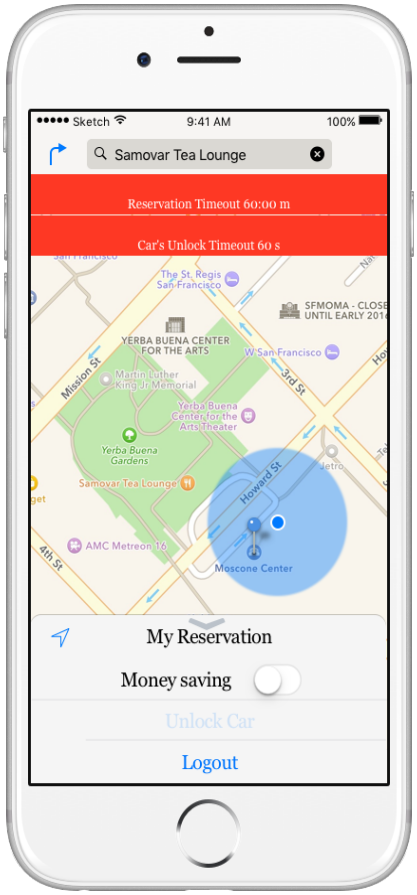

Figure 3.3: *car unlocked*

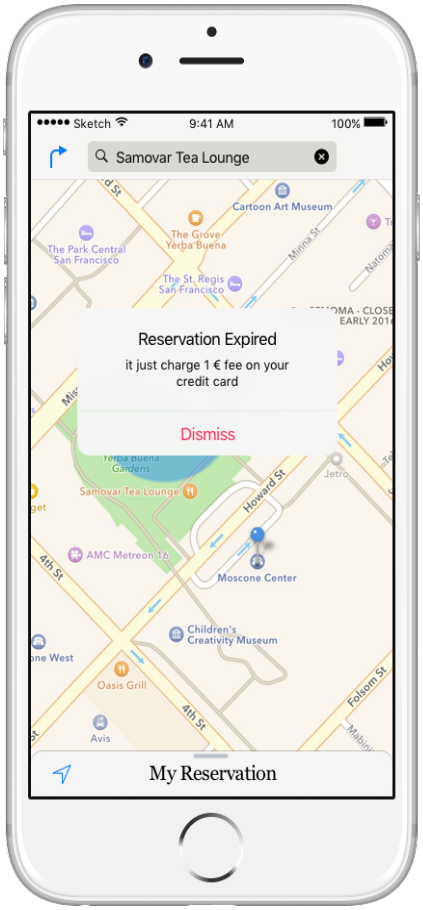

Figure 3.4: *reservation expired*

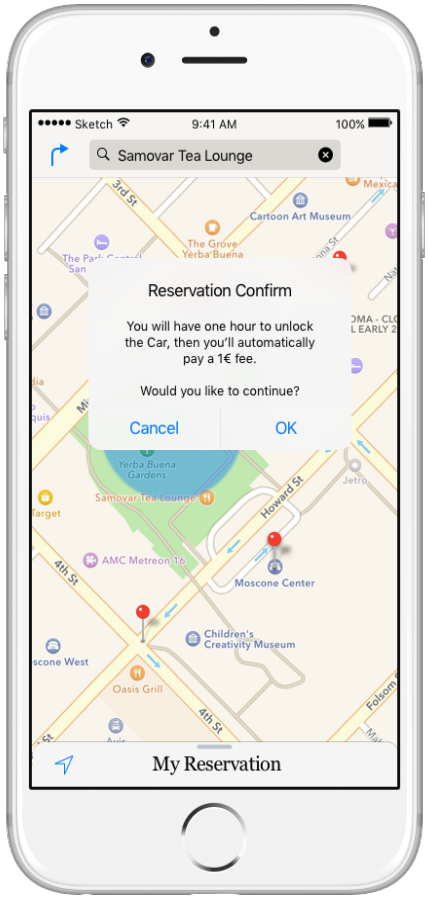

Figure 3.5: *reservation confirm*

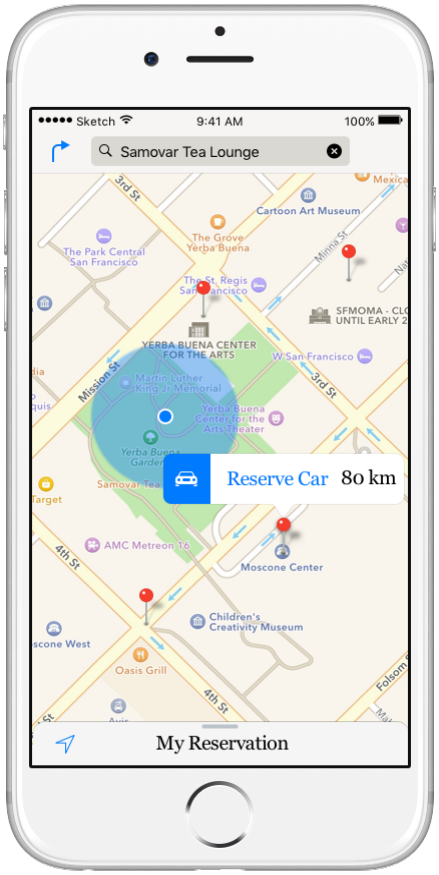

Figure 3.6: *reserve car*

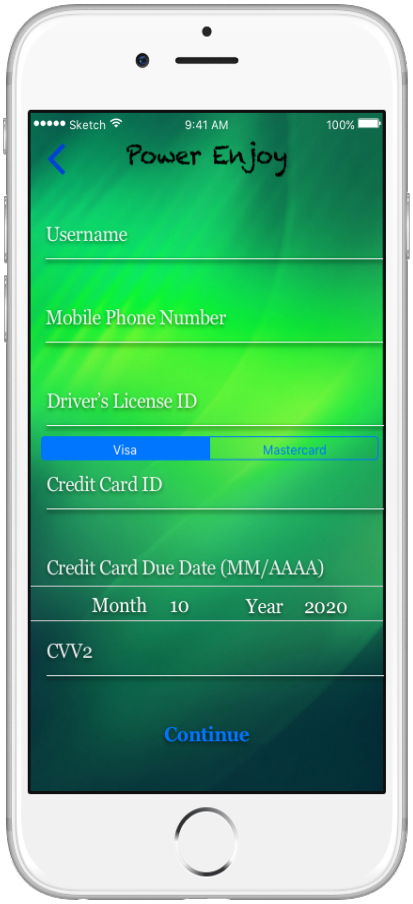

Figure 3.7: *registration page*

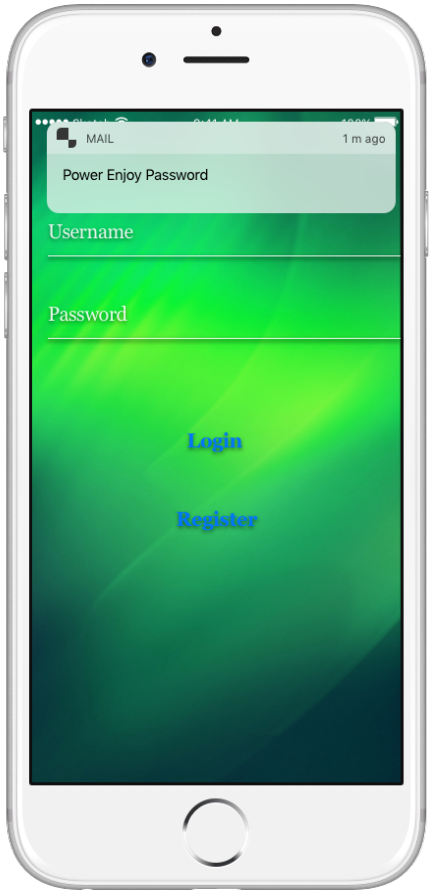

Figure 3.8: *login page*

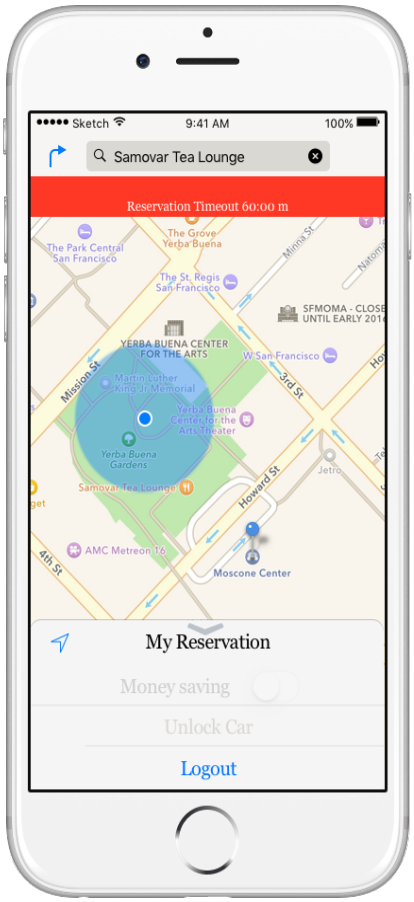

Figure 3.9: *reserved car, but not reached*

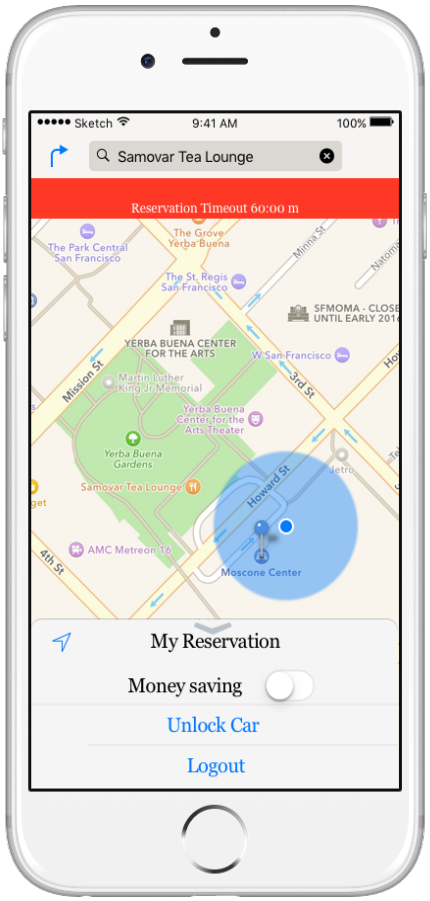

Figure 3.10: *reserved car and reached*

3.2 Functional Requirements

3.2.1 Goals

Goal ID	G1
Goal Description	Ensure system's accessibility
Goal Requirements	• The system must prevent guests from accessing any service before being registered or logged in • The system must recognize already registered user • The system must allow new user's registration • The system must allow user's login • The system must check data correctness (including payment method validity) • If user is new and data are correct, system must provide a password to the user

Goal ID	G2
Goal Description	Supervisors must be able to check cars' status
Goal Requirements	• The system must be able to check every (car's remaining power) • The system must be able to check every car's position • The system must be able to check if a car is in use • The system must be able to check if a car is reserved • The system must be able to check every car's "availability state" • The system must be able to check how many passengers are in the cars • The system must be always able to communicate with each car • The system must keep the information of every car updated

Goal ID	G3
Goal Description	Supervisor should be able to dispatch "recharge on site" correctly
Goal Requirements	• The system must notify the supervisor if a car is left with less than 20% of the battery and is not plugged into a power grid

Goal ID	G4
Goal Description	Supervisor should be able to dispatch "car recovery" correctly
Goal Requirements	• The system must notify the supervisor if a car is left outside a safe area for more than 5 hours

Goal ID	G5
Goal Description	Guarantee the correctness of each car's "availability state"
Goal Requirements	• The system must consider a car "unavailable" if it has low battery (<20%) • The system must consider a car "unavailable" if it has already been reserved by a user • The system must consider a car "unavailable" if it is not parked in a safe area • The system must consider a car "available" in any other case • The system must consider a reservation expired after 60 minutes if the car reserved isn't used • The system must consider a reservation expired 60 seconds after the car reserved is parked in a safe area and the user switch off the car and exit • The system must consider a reservation expired if the reserved car remains parked outside a safe area for more than 5 hours

Goal ID	G6
Goal Description	Allow user to find available cars within a certain distance from a specified place
Goal Requirements	• The system must be able to detect the user's location according to the userâ€™s device's GPS. • The system must be able to detect cars' location according to the cars' GPS. • The system must be able to detect a specific location according to the address provided by the user • The system must be able to determinate the distance between available cars and the indicated position • The system must show to the user the position, on the app's map, of the available cars that are within 1500 meters from the indicated position

Goal ID	G7
Goal Description	Allow user to reserve a single car
Goal Requirements	• The system must allow the user to select a car among the ones that are showed after the search • The system must show the user the "reserve" button after he has selected a car • The system must show the user the estimated autonomy of the selected car • The system must not show the reservation button if no car has been selected first • The system must reserve the selected car for the user after he clicks the "reserve" button • The system must prevent a user to reserve more than one car at a time

Goal ID	G8
Goal Description	Discourage fake and too long reservation
Goal Requirements	• The system must notify the user about the fee he will pay if he won't use the car that he is reserving within 60 minutes • The system must emit a payment request of 1 EUR to the credit card of the user who has reserved a car and did not use it within 60 minutes

Goal ID	G9
Goal Description	Allow the user who reserved the car to see information about his reservation
Goal Requirements	• The system must allow the user who reserved a car to access the "My reservation" page • The system must show the "unlock" button in the "My reservation" page • The system must show when the reservation will expire if the car reserved was not used yet • The system must show usage time if the car has already been ignited • The system must show the actual cost of the trip if the car reserved was not used yet • The system must show the total amount of the bill as soon as the reservation expired

Goal ID	G10
Goal Description	Allow only the user who reserved the car (and his passenger) to access it
Goal Requirements	• The system must be able to check the position of the user • The system must allow the user to click the "unlock" button after he has reserved a car • The system must not accept request of unlock if the user is more than 100 meters away from the car • The system must unlock the car when he receives the unlock request from the user • The system must lock the car if no user enters the car within 60 seconds from the unlock request • The system must lock the car if the user and all the passengers exit the car

Goal ID	G11
Goal Description	Guarantee the correctness of the "cost of the trip"
Goal Requirements	• The system must be able to check when the car's engine ignites • The system must start count the minutes of car's usage as soon as the engine ignites • The system must stop count the minutes of car's usage as soon as the car's reservation is considered expired (see G5) • The system must calculate the "cost of the trip" based on the formula: Cost of the trip = (timeOfExpiration + - timeOfFisrtIgnition) $*$ (cost per minute) (see class and sequence diagrams)

Goal ID	G12
Goal Description	Guarantee the correctness of the "virtuousness coefficient"
Goal Requirements	• The system must be able to recognize how many passengers are in the car • The system must consider the virtuousness coefficient's initial value equal to 1 • The system must subtract "0.1" to the "virtuousness coefficient" if the user has shared the trip with at least 2 other passengers • The system must subtract "0.2" to the "virtuousness coefficient" if the user left the car in a safe area with at least 50% of its battery power • The system must subtract "0.3" to the "virtuousness coefficient" if the user cares to plug the car into a power grid within 60 seconds he parked the car in a safe area • The system must add "0.3" to the "virtuousness coefficient" if the user left the car with less than 20% of its battery power • The system must add "0.3" to the "virtuousness coefficient" if the user left the car at more than 3 km from the nearest power grid station

Goal ID	G13
Goal Description	Guarantee the correctness and the payment of the final bill
Goal Requirements	• The system must calculate the amount of the final bill total using the formula: amount = (cost of the trip) $*$ (virtuousness coefficient) • The system must emit a payment request of the amount of the final bill using the payment method of the last user, after 60 seconds that the last reservation has expired

Goal ID	G14
Goal Description	Support the users saving money
Goal Requirements	• The system must allow the user to check the "money saving" option when he reserves a car • The system must allow the user to insert his destination after he checks the money saving option • The system must be able to recognize the place entered by the user • The system must know the availability of each power plug after 60 seconds that the last reservation has expired • The system must calculate the nearest safe area to the user's destination, with free power grids where to attach the car

3.2.2 Scenarios

Scenario ID	S1
Title	Deal with the strike
Description	Luca should go to class this afternoon, but unfortunately, today there is a strike of transport. His university is on the opposite part of the city, so he decides to try the new car-sharing service "PowerEnJoy". Since it's the first time he tries the service, first he should download the app and register to the system; after filling in his own personal data, including the payment information, he clicks on the submit button and, after a few seconds, he receives a message with his password. Now he can start looking for a car near to him.

Scenario ID	S2
Title	Friends in saving
Description	Marco, an expert user of "PowerEnJoy", has gone to see a concert with his roommates Mario and Matteo, and now they want to come back home. Due to the late hour, the public transportation is no more available and, as the evening was rather expensive, they aim to spend as little as possible. Marco decides to use the "PowerEnJoy" service and, after he found and reserved a car near to them, he checks the "Money saving" option. When they get into the car, they set up their destination and the system calculates the most convenient place to leave the car. When they arrive at their destination, they will have to walk a bit, but they will have saved a lot.

Scenario ID	S3
Title	A busy businessman
Description	William is a businessman always in a hurry; he has just arrived to his office, but he already knows that, as soon as he will finish the morning's meeting, he will have to go to the opposite part of the city for urgent commitments. William saw a "PowerEnJoy" car parked at a few meters from his office, so he thinks that he can save time using the car-sharing service instead of wait for a taxi after the meeting. William then register quickly to the service without paying much attention to all warnings and book the car. Unfortunately, the meeting dwells and William's reservation expires and the system charges him 1 EUR; when he leaves the office, he cannot get in the car because it was booked by another user, forcing William to call a taxi.

Scenario ID	S4
Title	Desperate housewife
Description	Laura went to the grocery store on foot, but when she exits the supermarket she realizes that it starts raining; she notices a "PowerEnJoy" car parked and, since his son has already registered her to the service to encourage her to use it, she decides to book the car to come back home without getting wet. Once she arrives at destination, the car is low, but Laura's first problem is to not get wet, so she looks for a park in a safe area as close as possible to her house, and she doesn't mind the warning concerning the fact she will pay more if she won't leave the car in a recharge park.

Scenario ID	S5
Title	A Long Road
Description	Yuri recently moved to a new built house so he needs some furniture. The only store he knows is an Ikea so he decides to go there, also if it is quite far. Having in mind to buy some initial stuff like pillow and sheets, he knows that he won't use the Home Delivery, as it would turn out in an extremely high fee for such small items. He thinks then to use a "PowerEnjoy"'s car. After the registration, he finds out that there's a car near his house, so he decides to reserve it. Then, he goes near the car and clicks on the "unlock" button in the "PowerEnJoy" app on his smartphone, he enters and starts driving. When he arrives at the IKEA, he finds out that there is no safe area where to park the car! Now he knows that he must do his errand quickly, because he will pay every minute for the car also if it's switched off, and after 5 hours his reservation will expire!

3.2.3 Use Case Diagrams

Figure 3.11: *PowerEnjoy Use Case Diagram*

3.2.4 Use Cases Description

Description use case 1

Title: User registration

Actor: Guest

Entry condition:

- The guest user is on the home page of the app on his mobile device

Flow of Events:

1. The guest clicks on "Register"
2. The system shows the form on screen
3. The guest inserts his name, mail, mobile phone number and the data of his payment method and then clicks on "Continue"
4. The system checks if the mail and phone number are syntactically valid, and if the payment method inserted is valid; if so, it generates the user's password and sends it to him via email
5. The system shows the form for the password insert
6. The guest inserts the password that he received and clicks on "continue"
7. The guest is now a customer registered to the service

Exit condition:

1. If the password inserted is correct, the guest is now a customer registered to the service and the system redirects the user to the page dedicated for searching a car
2. If the password inserted isn't correct, the system redirects the user to the home page

Exception:

- If the password inserted isn't correct, the system redirects the user to the home page
- One or more fields are not well-formed
- The payment method results not valid

Description use case 2

Title: User registration

Actor: Guest

Entry condition:

- The guest user is on the home page of the app on his mobile device

Flow of Events:

1. The guest user is on the home page of the app on his mobile device

Exit condition:

1. The guest user is on the home page of the app on his mobile device

Exception:

- The guest user is on the home page of the app on his mobile device

Description use case 3

Title: Dispatch recharge on site

Actor: Supervisor

Entry condition:

- The system notices that the remaining power of the battery of a car has fallen to 20%

Flow of Events:

1. The system waits until the reservation for that car has expired
2. The system checks if the car has been plugged into a power grid
3. If this hasn't happened, the system shows to the supervisor the "dispatch recharge on site" button
4. The supervisor clicks on the "dispatch recharge on site" button
5. The information about the car's position are sent to a worker

Exit condition:

- The worker go to the car
- The car has been plugged into a power grid

Exception:

- There are no exceptions for this use case

Description use case 4

Title: Look for a car

Actor: User

Entry condition:

- The user is on the page dedicated to search cars

Flow of Events:

1. The user clicks on the "search a car" button
2. The system recognizes the position of the user through his device's GPS
3. The system calculates the distance between the position of the user and the position of each available car
4. The system sends to the user the position of the available cars whose distance from the user is less than 1500 meters

Alternative Flow:

1. The user inserts an address in the text box
2. The user clicks on the "search a car" button
3. The system recognizes the position of the address inserted by the user
4. The system calculates the distance between the position of the address inserted and the position of each car
5. The system sends to the user the position of the available cars whose distance from the inserted address is less than 1500 meters

Exit condition:

- The system shows on the app's map, on the user's device, an icon for each car found

Exception:

- There are no available cars nearby the position indicated

- The address inputted is not valid

Description use case 5

Title: Reserve a car

Actor: User

Entry condition:

- The user searched a car and the system has found at least one available

Flow of Events:

1. The user select a car among the ones shown on his app by clicking on its icon
2. The system shows the user the "reserve" button
3. The user clicks on the "reserve" button
4. The system creates the user's reservation
5. The system change the availability state of the reserved car into "unavailable"

Exit condition:

- The system informs the user about the succeed of the reservation

Exception:

- The car that the user is trying to reserve has already been reserved by another user in the meantime
- The user has another active reservation

Description use case 6

Title: Unlock a car

Actor: User

Entry condition:

- The user has reserved a car
- The user is on the "Your reservation" page

Flow of Events:

1. The user clicks on the "unlock" button
2. The system recognizes the position of the user through his device's GPS
3. The system calculates the distance between the position of the user and the position of the car

Exit condition:

- If the user is within 100 meters from the car, the system unlocks the car and notifies the user
- If he user isn't within 100 meters from the car, the system notifies that he needs to approach the car

Exception:

- If no one get in the car within 60 seconds from the unlock, the system relocks the car

Description use case 7

Title: Pay the bill

Actor: User

Entry condition:

- The user's reservation has expired

Flow of Events:

1. The system check why user's reservation expired
2. If the reservation has expired for inactivity, the total amount of the bill is 1 EUR
3. In the other case, the total amount of the bill is calculated in the standard way

Exit condition:

- The system emits the payment request

Exception:

- If no one get in the car within 60 seconds from the unlock, the system relocks the car

3.2.5 Sequence Diagrams

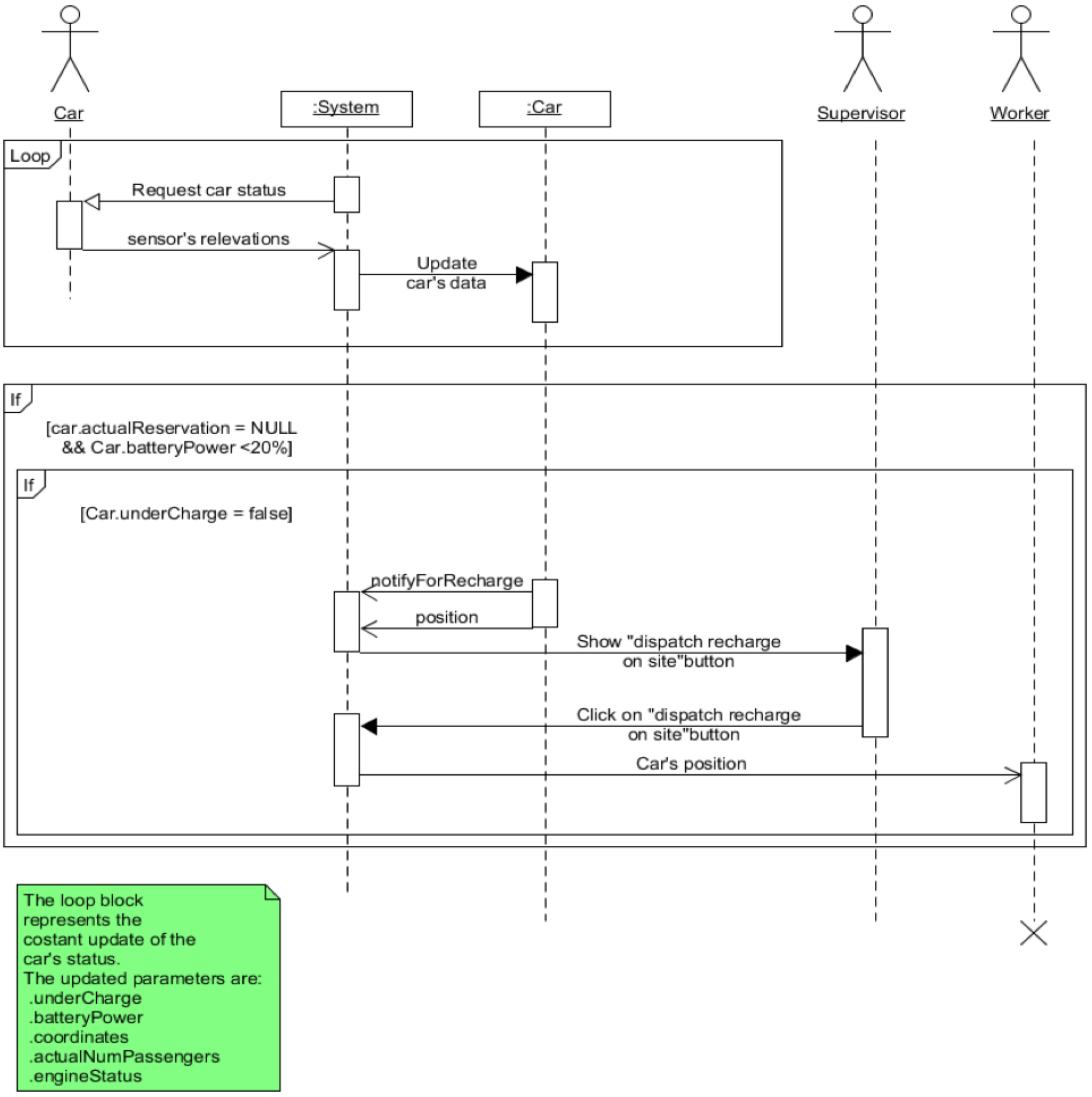

Figure 3.12: *Sequence Diagram Dispatch Recharge On-Site*

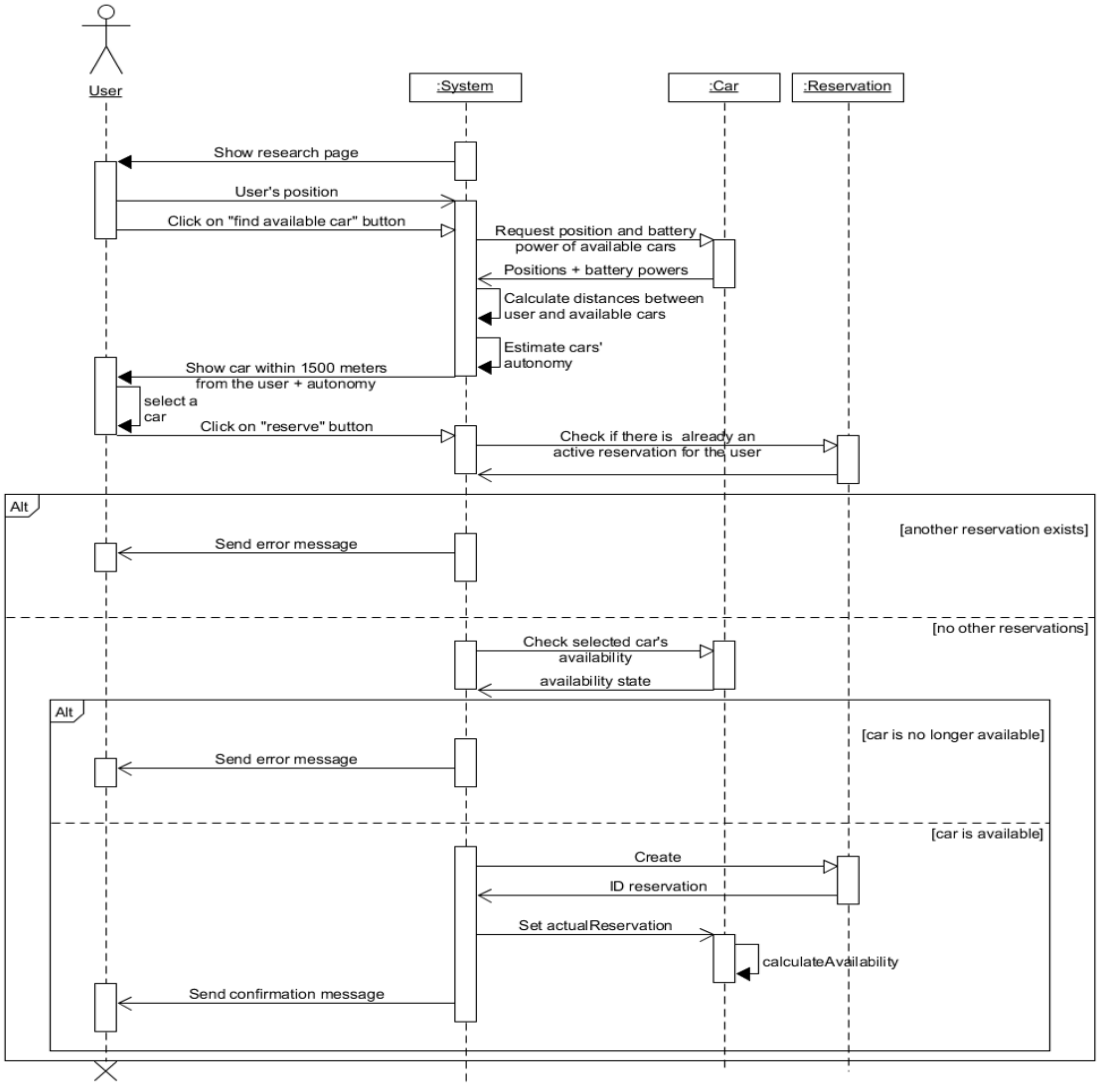

Figure 3.13: *Sequence Diagram Look For And Reserve A Car*

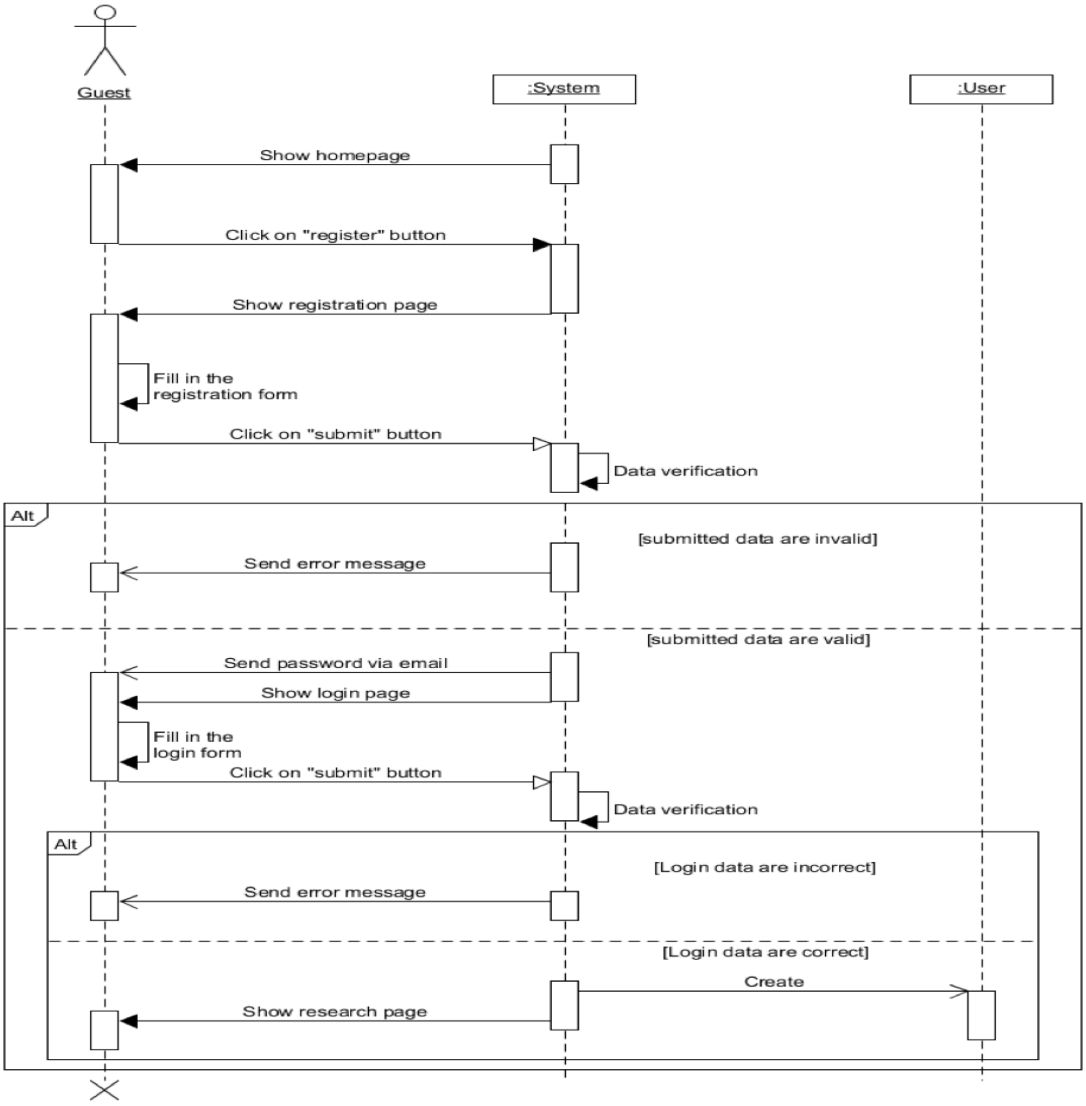

Figure 3.14: *Sequence Diagram Registration*

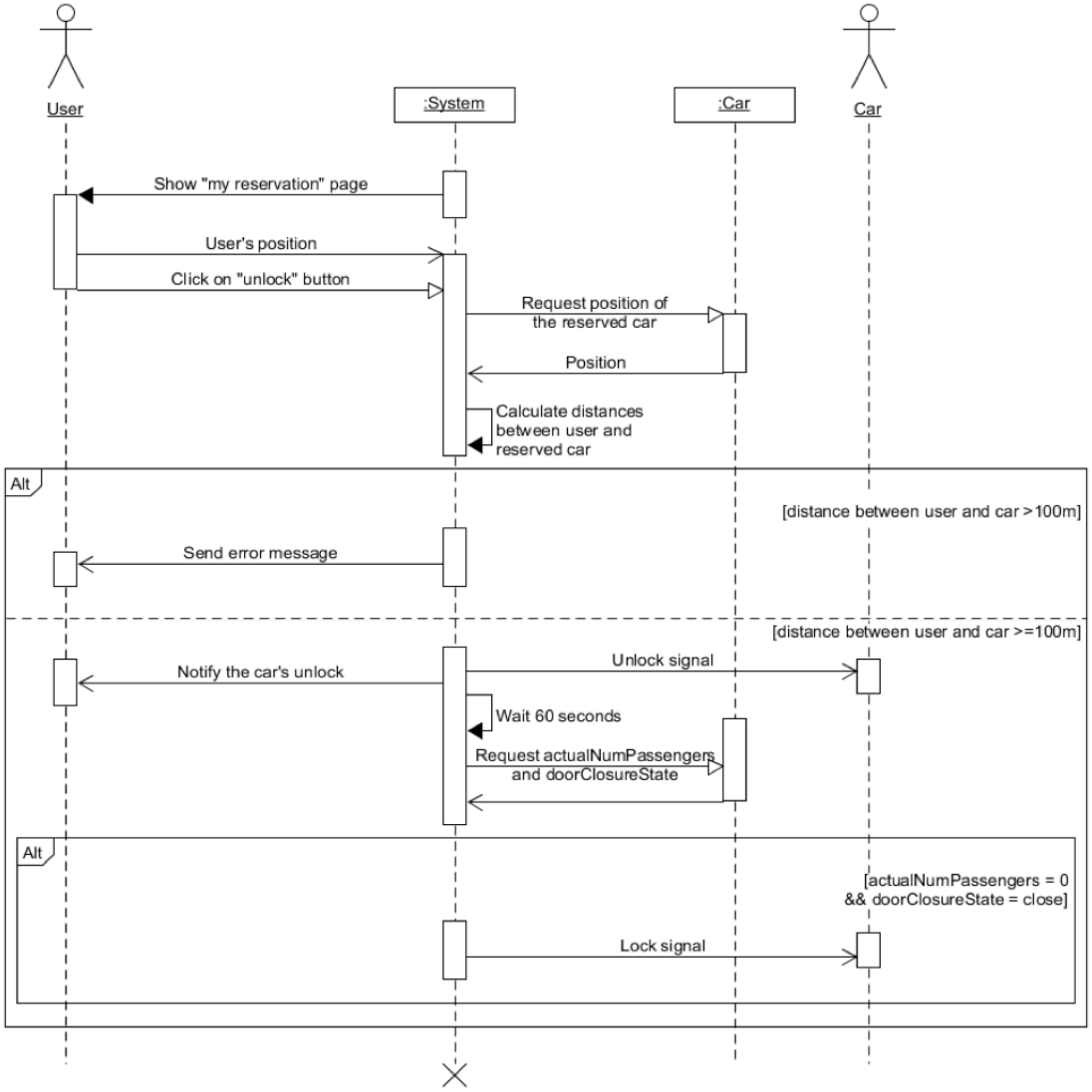

Figure 3.15: *Sequence Diagram Unlock TheReserved Car*

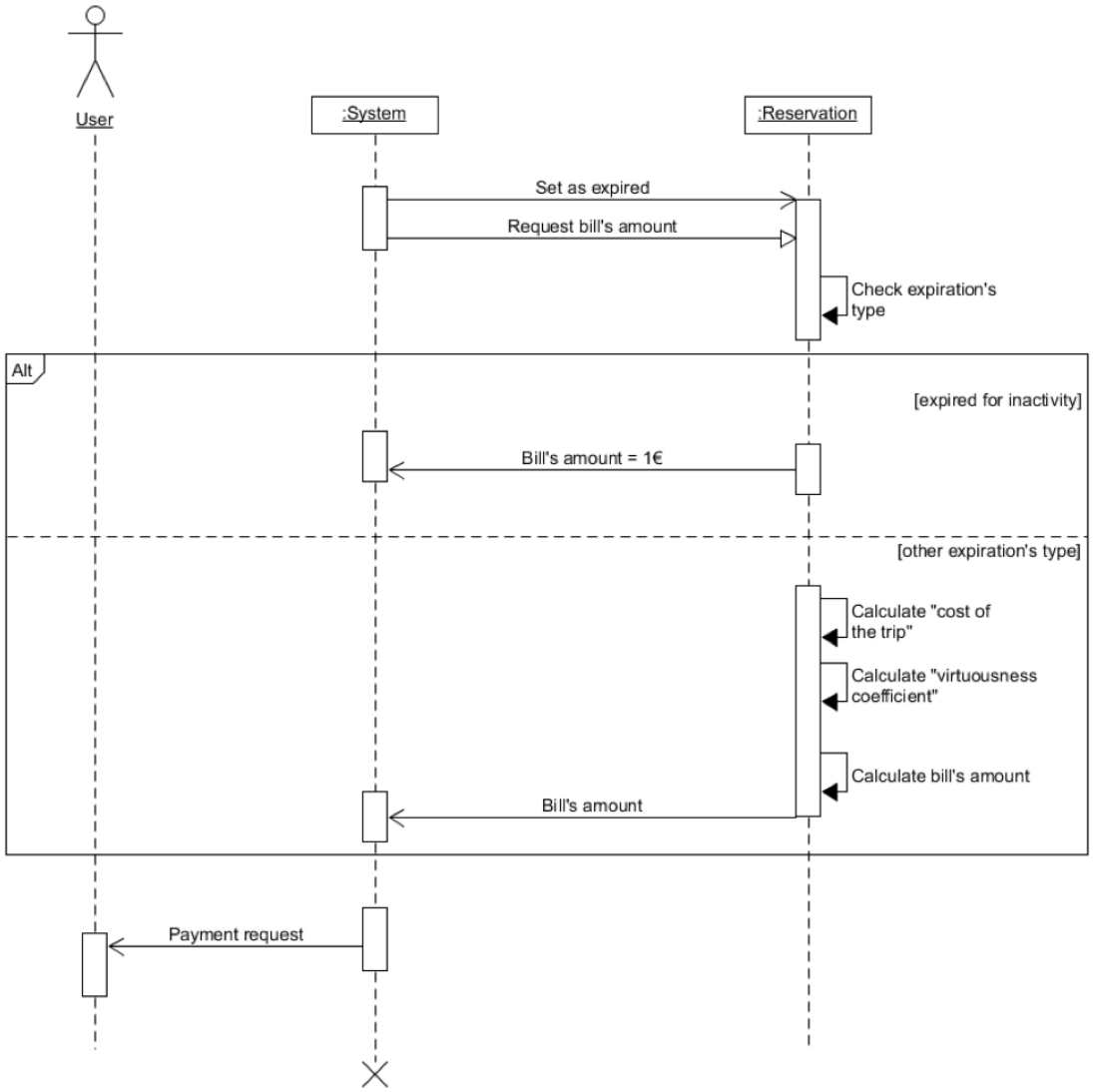

Figure 3.16: *Sequence Pay The Bill*

3.2.6 Alloy

Model

```
//////////////////ENUMS//////////////////

enum CarStatus {
        AVAILABLE,
        UNAVAILABLE
}

enum CarEngine {
        ON,
        OFF
}

enum LotStatus {
        EMPTY,
        USED
}

enum ReservationStatus {
        EXPIRED,
        ACTIVE
}

enum BatteryStatus {
        FULL,
        MEDIUM,
        LOW,
```

```
    VERYLOW
}

//////////////////PREDICATES//////////////////

sig ValidString{}

sig Supervisor {
    manages: set Car
}{}

sig RegisteredUser {
    name: one ValidString,
    creditCard: one Int,
    position: one Position
}{
    #creditCard > 0
}

sig ReservationDB{
    contains: set Reservation
}{
    #ReservationDB = 1
}

sig Reservation {
    reservationID : Int,
    bookedBy: one RegisteredUser,
```

```
    relatedTo: one Car,
    reservationStatus: one ReservationStatus
}{
    reservationID > 0
}

sig Position {
    position_x: one Int,
    position_y: one Int
}

sig Car {
    carID: Int,
    parkedInto: lone ParkingLot,
    position: one Position,
    carStatus: one CarStatus,
    carEngine: one CarEngine,
    battery: one BatteryStatus
}{
    carID >= 0
}

sig SafeArea {
    contains: set ParkingLot
}
{
    #contains > 0
}
```

```
sig ParkingLot {
        safeArea : one SafeArea,
        position : one Position,
        status: one LotStatus
}

sig CHParkingLot extends ParkingLot{}

//////////////////FACTS//////////////////

// There's at least a Safe Area in the world
fact atLeastASafeArea{
        #SafeArea > 0
}

// There's at least a car in the world
fact atLeastACar{
        #Car > 0
}

// There's only a Supervisor
fact thereIsOnlyASupervisor{
        #Supervisor = 1
}

// Every car is managed by the Supervisor
fact everyCarIsManaged{
```

```
        all c : Car | c in Supervisor.manages
}

// Every reservation is stored, even the Expired ones
fact theReservationDBContainsAllTheReservations{
        all r: Reservation | one rDB: ReservationDB | r in rDB.contains
}

// There aren't duplicated Cars
fact noDuplicatedCar {
        no car1 , car2 : Car | (car1 != car2) && ( car1.carID = car2.carID)
}

// No cloned reservations
fact noClonedReservations {
        no reservation1 , reservation2 :
        Reservation | (reservation1 != reservation2) &&
        (reservation1.reservationID = reservation2.reservationID)
}

// different parkingLot must have different positions
fact PLUnity{
        no pl1, pl2 : ParkingLot | (pl1 != pl2) && (pl1.position = pl2.position)
}

// Different car must have different positions
fact correctCarPosition{
        no c1, c2 : Car | (c1 != c2) && (c1.position = c2.position)
```

```
}

// Two car can't be parked at the same ParkingLot
fact noAbusedParkingLot {
        no car1,car2 : Car | (car1 ! = car2) && ( car1.parkedInto = car2.parkedInto)
}

// Check for correct Pl's statuses
fact PLStatusesconsistency{
        all pl : ParkingLot |
        (no c: Car | (c.parkedInto = pl)) => (pl.status ! = USED)
        else (pl.status = USED)
}

// A parked car must have the same position of a parkingLot
fact carAndLotInTheSameZone{
        all c : Car | #c.parkedInto > 0 => ( c.position=c.parkedInto.position)
}

fact carNotParkedHasNoSamePositionOfPL {
        all c : Car, pl:
        ParkingLot | #c.parkedInto = 0 => (c.position ! = pl.position)
}

// Containment relation must be bidirectional
fact bidirectional1{
        all s: SafeArea, pl: s.contains | pl.safeArea = s
}
```

```
fact bidirectional2{
        all pl: ParkingLot, s: pl.safeArea | pl in s.contains
}

// A user can't have two "Active" reservations
fact{
        no r1, r2: Reservation | ( (r1! =r2) &&
        (r1.reservationStatus = ACTIVE) &&
        (r2.reservationStatus = ACTIVE)) &&
        (r1.bookedBy = r2.bookedBy)
}

// A car should be related for just one "ACTIVE" reservation at a time
fact{
        no r1, r2: Reservation | ((r1 ! = r2) &&
        (r1.reservationStatus = ACTIVE) &&
        (r2.reservationStatus = ACTIVE)) &&
        (r1.relatedTo = r2.relatedTo)
}

// A car should be "Available" only if it's parked in a safe area
fact CarAvailableOnlyIfInSafeArea{
        no c: Car | c.parkedInto = none && c.carStatus = AVAILABLE
}

// A car whose engine is on, must be connected to an active reservation
fact WorkingCarMeansActiveReservation{
```

```
        all c : Car | some r : Reservation |
        c.carEngine = ON => r.relatedTo = c &&
         r.reservationStatus = ACTIVE
}

// If a car is being used, should be "Unavailable"
fact carStatusConsistency{
        all r : Reservation |
        (r.reservationStatus = ACTIVE)
        => (r.relatedTo.carStatus = UNAVAILABLE)
}

// All car with low battery and not already in use, are not availabe for book-
ing
fact batteryConsistency{
        all c : Car | (c.battery = VERYLOW)
         => (c.carStatus = UNAVAILABLE)
}

// All car parked in a safe area,
with not empty battery and not booked, should be available
fact batteryConsistency2{
all c : Car | c.parkedInto ! = none && c.battery ! = VERYLOW &&
        ((Reservation :> relatedTo.c) &
        (Reservation :> reservationStatus.ACTIVE)) = none
        => (c.carStatus = AVAILABLE)
}
```

```
//////////////////ASSERTIONS//////////////////

pred show{}

// A parking lot whose position coincides
with the position of a car, should not be marked as empty
assert plConsistency{
        no c: Car, pl: ParkingLot | (c.position = pl.position && pl.status = EMPTY)
}

check plConsistency for 10

// A car whose position does not coincide
with any parking lot's position should be UNAVAILABLE
assert carOutsidePLUnavailable{
        no c: Car | c.position& ParkingLot.position = none &&
        c.carStatus= AVAILABLE
}

check carOutsidePLUnavailable for 10

// Active reservations should refers to different cars
and different users
assert differencesBetweenActiveRes{
        no r1, r2: Reservation | (r1 ! = r2) &&
        (r1.reservationStatus = ACTIVE) &&
        (r2.reservationStatus = ACTIVE) && ((r1.bookedBy = r2.bookedBy)
         || (r1.relatedTo = r2.relatedTo))
```

```
}

check differencesBetweenActiveRes for 10

// All car with not empty battery, not booked
and parked, should be available
assert chargedCarAvailability{
        no c: Car | c.parkedInto != none && c.battery! =VERYLOW &&
        Reservation :> relatedTo.c &
        Reservation :> reservationStatus.ACTIVE = none &&
         c.carStatus = UNAVAILABLE
}

check chargedCarAvailability for 10

// A car whose engine is working should not be available
assert noAvailableWorkingCar{
        no c: Car | c.carEngine = ON && c.carStatus = AVAILABLE
}

check noAvailableWorkingCar for 0

//////////////////////

/*
assert noCarAvailable{
        no r1, r2, r3: Car | (r1 != r2) && (r2! = r3) && (r1 != r3) &&
        (r1.carStatus=AVAILABLE) && (r2.carStatus=AVAILABLE) &&
```

```
        (r3.carStatus=AVAILABLE)
}

check noCarAvailable blue for 5

assert noReservationActive{
        no r1, r2, r3: Reservation | (r1 ! = r2) && (r2! =r3) &&
        (r1 ! = r3) && (r1.reservationStatus = ACTIVE) &&
        (r2.reservationStatus = ACTIVE) && (r3.reservationStatus = ACTIVE)
}

fact noReservationActive for 5
*/

run show for 5
```

Result

Here is the result of Alloy Analyzer for each assert,predicate,fact and check.

```
Executing "Check plConsistency for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35682 vars. 2240 primary vars. 76892 clauses. 63ms.
  No counterexample found. Assertion may be valid. 78ms.

Executing "Check carOutsidePLUnavailable for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35594 vars. 2230 primary vars. 76321 clauses. 63ms.
  No counterexample found. Assertion may be valid. 156ms.

Executing "Check differencesBetweenActiveRes for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35974 vars. 2240 primary vars. 77845 clauses. 59ms.
  No counterexample found. Assertion may be valid. 94ms.

Executing "Check chargedCarAvailability for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35653 vars. 2230 primary vars. 76441 clauses. 46ms.
  No counterexample found. Assertion may be valid. 79ms.

Executing "Check noAvailableWorkingCar for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35386 vars. 2230 primary vars. 75944 clauses. 46ms.
  No counterexample found. Assertion may be valid. 47ms.

Executing "Run show for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35260 vars. 2220 primary vars. 75722 clauses. 63ms.
  Instance found. Predicate is consistent. 339ms.

6 commands were executed. The results are:
  #1: No counterexample found. plConsistency may be valid.
  #2: No counterexample found. carOutsidePLUnavailable may be valid.
  #3: No counterexample found. differencesBetweenActiveRes may be valid.
  #4: No counterexample found. chargedCarAvailability may be valid.
  #5: No counterexample found. noAvailableWorkingCar may be valid.
  #6: Instance found. show is consistent.
```

Figure 3.17: *Alloy Result*

Generated World Examples

Here are presented two generated worlds, according to the model specified in Alloy.

Figure 3.18: *Example 1*

Figure 3.19: *Example 2*

Chapter 4

Notes

4.1 Revision Notes

Version 1.0 - First Release

4.2 Used Tools

The tools we used to create this RASD document are:

- **Alloy Analizer 4.2:** to prove the consistency of our model.
- **Sketch:** to design mockups
- **Github:** for version controller and to manage the phase of the project
- **Textpad:** to design RASD document with LaTeX
- **UMLet:** to design UML diagrams

4.3 Hours of work

Date	Marco (h)	Angelo (h)	Gabriele (h)
22/10/2016	1		1
23/10/2016	1		2
24/10/2016	1		0.5
25/10/2016	1	2	
26/10/2016	1		3
27/10/2016	2		1
29/10/2016	1		1
30/10/2016	1		2
31/10/2016	1		2
02/11/2016	1	3	1
03/11/2016	1		3
05/11/2016	2.5		2.5
06/11/2016	1	2	3
07/11/2016	2		2
08/11/2016	8	2	4
09/11/2016	4	6	2
10/11/2016	0.5	3	
11/11/2016		1	
12/11/2016		3	
13/11/2016	1.5	9	
TOTAL	31.5	31	30