

Requirements Analysis and Specifications Document

Software Engineering 2

November 13, 2016



POLITECNICO
MILANO 1863

Authors:

Claudio Salvatore	ARCIDIACONO	Matr 879109
Antonio	DI BELLO	Matr 878786
Denis	DUSHI	Matr 879115

Contents

1	Introduction	2
1.1	Description of the given problem	2
1.2	Goals	2
1.3	Domain assumption	2
1.4	Glossary	3
1.5	Synonymous	3
1.6	Interfaces to external applications	3
1.7	Reference Documents	4
2	Actors	5
3	Requirements	6
3.1	Functional requirement	6
3.2	Non functional requirement	9
3.2.1	View available car information in mobile application	9
3.2.2	View map on the website	10
3.2.3	Registration form	11
3.2.4	Insert pin on the car display	12
3.3	The world and the machine	12
4	Scenarios	13
4.1	Registration Scenario	13
4.2	Log in and display car information Scenario	13
4.3	Reservation Scenario	13
4.4	Opening / Rental scenario	13
4.5	Ecology Scenario	13
4.6	Reporting dirty car Scenario	13
4.7	Saving Mode Scenario	14
5	UML models	15
5.1	Use Cases Description	15
5.1.1	Register to PowerEnJoy	15
5.1.2	Log-in	15
5.1.3	Display a map information	16
5.1.4	Reserve a car	16
5.1.5	Open a car	17
5.1.6	Start the rent	17
5.1.7	End the rent	18
5.1.8	Plug the car	18
5.1.9	Report damaged/dirty car	18
5.1.10	Select money saving option	19
5.2	Use Case Diagram	20
5.3	Class Diagram	21
5.4	Sequence Diagrams	22
6	Alloy	25
6.1	Code	25
6.2	Results	29
6.3	Generated world	30
7	Appendix	31
7.1	Software and tool used	31
7.2	Hours of work	31
7.3	Changelog	32

1 Introduction

1.1 Description of the given problem

We will project and implement PowerEnJoy, which is a digital management system for a car-sharing service that exclusively employs electric cars. The system that will be developed has to allow users to register and log in to the service, to see which car are available and where the cars can be parked. In addition to the basic functionality of a car-sharing service the system has to incentive virtuous behaviors of the users, like plugging the car to a charging station or leaving the car with enough charge, in order to have as much as possible a self-sustainable service.

1.2 Goals

- G1 A client can register to the system by providing an e-mail, valid payment information and a photo of his driving license.
- G2 A client can log in to system.
- G3 Allows clients to obtain information about available cars(position and remaining charge), safe areas(boundaries) and charging stations (position and available plugs).
- G4 Allows clients to reserve a car that fits most their needs.
- G5 A client can start the rent opening a car that has reserved previously when he/she is in the near by.
- G6 During the rent a client can display the amount of money charged.
- G7 Guarantee as many available cars as possible encouraging clients to have a virtuous and eco-friendly behavior applying fees and discounts.
- G8 Allows clients to end the rent and leave the car in any safe area.
- G9 Clients can report eventual damages made by users that have driven the car before.
- G10 Clients can set the money saving option.

1.3 Domain assumption

We suppose that these properties hold in the analyzed world :

- All cars have a stable GPS signal.
- Clients shares their position when they use the application.
- PowerEnJoy always has updated data about all the cars and the power stations.
- The power grid always guarantees electric power.
- The only way to get in a car is by the doors.
- The company cars have at least 4 seats.
- The streets are correctly mapped in the system.
- All cars have stable internet connection.
- The charging sensor reflects the real charge of the car in real-time.
- The real charge of parked cars remains the same over time.
- Charging stations are not in the same GPS position.
- There aren't two clients with the same personal data.

1.4 Glossary

Guest: is an user that is not registered yet to PowerEnJoy.

Client: is an user that has completed successfully the registration procedure.

Logged client: is a client that has logged into PowerEnJoy.

Blocked client: is a client that is not allowed to log in to system because he has some debits with PowerEnJoy.

User: a client or a guest.

Virtuous behavior: the client has a "Virtuous behavior" if he performs one or more of the following actions:

- transport at least other two passengers into the car.
- leaves the car with less than 50% of the battery empty.
- plugs the car into the power grid of a charging station.

Rent: defines the period of time that starts when the logged client opens the car and ends when the door of the car are locked due to a "end rent" request of the logged client.

Pin: a personal sequence of four numbers given by the system after the registration, necessary to start the engine.

Map Information: denotes the following information:

- location of available cars.
- location of charging areas.
- for each charging area the number of available charging spots.
- boundaries of safe areas.

Not valid data: syntactically incorrect data (e.g. mail not in the format local-part@domain).

Credentials: email and password used to log into PowerEnJoy.

Valid Credentials: email and password related to a registered user.

Car information Menu: displayed when a client selects a car on the map. In this menu are displayed the information about the car (remaining charge percentage and license plate number) and a button that can be used to reserve the car.

Safe Areas: areas in which is permitted to the client to end the rent and leave the car.

Valid Payment Information: Credit or prepaid card with at least a minimum amount of money.

1.5 Synonymous

- rental: rent
- charging area: charging station
- client : logged client. Sometimes an action can be performed only by a client that has logged previously. We will use client instead of logged client when this distinction is not ambiguous.

1.6 Interfaces to external applications

The PowerEnJoy system interacts with the following external systems:

- **Operators system:** to share with operators information about the rental chronology, the cars position, remaining charge and reports of damaged/dirty car for maintenance and legal purposes.
- **Civil registry office:** to obtain user profile information from ID number.
- **Transport authority:** to obtain user profile information from license number.
- **Payment gateway:** to verify the validity of the payment information and to charge the clients.

1.7 Reference Documents

- Specification Document: Assignments AA 2016-2017.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- Examples documents:
 - MeteoCal RASD example2.pdf
 - RASD sample from Oct. 20lecture.pdf

2 Actors

- **Guest:** this user hasn't registered to PowerEnJoy yet but he is allowed to view the map with all the information in the website or in the mobile application. To access to other functionality, like reservation, this type of user has to complete the registration procedure.
- **Client:** A Guest that performs correctly a registration procedure becomes a Client. The Client can log in to the PowerEnJoy platform using the e-mail provided during registration and password provided by system.
- **Logged Client:** Once the Client has logged into PowerEnJoy platform he is considered as a Logged Client. A Logged Client can still view the map but he can also reserve a car, open the reserved car, start and end the rent. During the rent he has the possibility to report a damaged or dirty car and to select the "money saving" option.

3 Requirements

3.1 Functional requirement

1. A client can register to the system by providing an e-mail, valid payment information and a photo of his driving license.
 - (a) The system must be able to check the validity of the payment information.
 - (b) The system must be able to check if the uploaded documents are in the accepted format.
 - (c) The system must be able to communicate with the civil registry office and transport authority in order to obtain user profile information from ID number and license number.
 - (d) The system must allow the registration only if ID ,license and personal information are related to the same person.
 - (e) The system must implement a retrieve password mechanism.
 - (f) The system should not allow to complete the registration process to an already registered user.
 - (g) The system should not allow different users to register with the same email.
2. A client can log in to system.
 - (a) The system should provide the log in function both in the app and the web application.
 - (b) The system should allow to rent a car only after the log in.
 - (c) The system must guarantee that none can access to a user profile without the valid credentials.
 - (d) The system must be able to check if the credentials inserted by the user are valid.
 - (e) The system must deny the access to blocked client.
3. Allows clients to obtain information about the position of available cars(position and remaining charge), safe areas(boundaries) and charging stations (position and available plugs).
 - (a) The system should acquire the GPS position of available cars.
 - (b) The system must offer a view of a map in which are displayed the position of the available cars and the charging stations and the boundaries of the safe areas.
 - (c) The system should update in real time the information about the charging stations and available cars.
 - (d) The system must detect the position of clients that have opened the application.
4. Allows clients to reserve a car that fits most their needs.
 - (a) The system should not allow a client to reserve more than one car simultaneously.
 - (b) The system must not show in on the map the car reserved.
 - (c) The system must delete automatically the reservation after 1 hour.
 - (d) The system must apply to the client a fee of 1 euro if the reservation lasts more than 1 hour.
 - (e) The system must allow to reserve only available cars that are shown in the map.
 - (f) The system should provide to the client the possibility to cancel the reservation.
 - (g) The system must set available a car when its reservation is canceled.
 - (h) A user that cancel a reservation can't reserve the same car within 15 minutes from the cancellation.
 - (i) The system must terminate the reservation when the client starts the rent.
5. A client can start the rent opening a car that has reserved previously when he/she is in the near by.
 - (a) The system should be able to know the user GPS position.

- (b) The system should permit a client to request the opening of a car only if there is an active reservation of that user for that car and the user GPS position is not more than 15 meters far from the car GPS position.
 - (c) The system should make possible to start the engine if and only if the user have inserted the correct pin.
 - (d) After the client authentication the system should provide a function to insert the destination and eventually select the saving mode option.
 - (e) The system must start the rent right after the car doors are unlocked.
 - (f) The car system must provide a step-by-step route information to reach the selected destination.
6. During the rent a client can display the amount of money charged.
- (a) The system must detect when the engine is on.
 - (b) The system must count the passed minutes to calculate the current charge.
 - (c) The system must refresh every minute the current cost of the rent, proportional to the passed minutes.
 - (d) The system must always show the current cost during the rent.
 - (e) The system must stop charging when the client turn off the engine, selects the "end rent" option and the car is parked in a safe area.
 - (f) The system must detect the moment when the client ends the rent.
7. Guarantee as many available cars as possible encouraging clients to have a virtuous and eco-friendly behavior applying fees and discounts.
- (a) The system must apply a discount of 10% on the last ride if it detects that during the rent the user had at least other two passengers into the car.
 - (b) The system must apply a discount of 20% on the last ride if the client leaves the car with less than 50% of the battery empty.
 - (c) The system must apply a discount of 30% on the last ride if the client plugs the car into the power grid of a charging station.
 - (d) The system must charge 30% more on the last ride if the client leaves the car at more than 3 KM from the nearest charging station.
 - (e) The system must charge 30% more on the last ride if the client leaves the car with a remaining charge of 20% or less.
8. Allows clients to end the rent and leave the car in any safe area.
- (a) The Display of the car system should provide a section that permits the user to end the rent.
 - (b) The system should allow a client to end the rent if and only if the client clicks the "end rent" button and the car GPS position is inside a safe area.
 - (c) After a minute from the end of the rent the system should lock the car doors.
 - (d) After 5 minutes from the end of the rent the system will charge the rent cost, which is proportional to the duration of the ride and takes into account eventual discounts.
 - (e) If the system tries to charge the user but the payment is unsuccessful the system blocks the log-in function of the user until the user doesn't absolve his debt.
9. Clients can report eventual damages made by users that have driven the car before.
- (a) The system must provide a section where the client can specify the entity of damage or the level of dirt.
 - (b) The system must interface with an external operator system.
 - (c) The system should be able to find the last user that used a car.

- (d) The system must allow the client to not start the rent and close the car, without applying any cost in case of damages.

10. Clients can set the money saving option.

- (a) The system must provide the possibility to select the "saving option" during the rent.
- (b) When the saving money option is enabled the system must provide information about the charging station, where one can leave and plug the car. This information is based on the destination selected, the availability of power plugs and the distribution of cars in the city.

3.2 Non functional requirement

Users can interact with our system through the mobile application, the website and the display located inside the car.

3.2.1 View available car information in mobile application

This mockup shows how the information of an available car (charge level and distance from the client) can be displayed in the mobile app. The logged client has the possibility to reserve the car by tapping on the "Reserve" button.

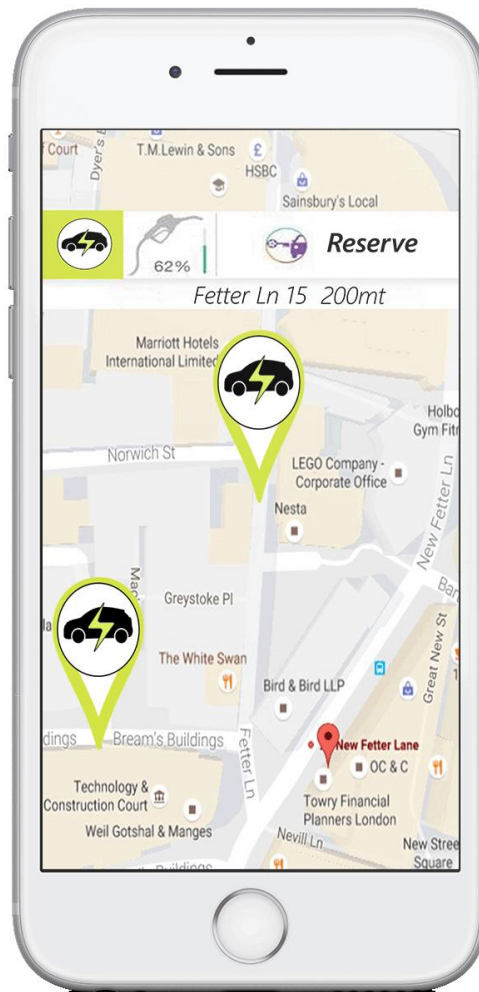


Figure 1: View available car information in mobile application.

3.2.2 View map on the website

This mockup shows the map that a generic user (guest or client) can see on the PowerEnJoy website. On the map are displayed the boundaries of the safe areas in which a client can leave the car and the charging stations with the number of available charging spots. The user can insert an address in order to see the available cars within a certain distance.

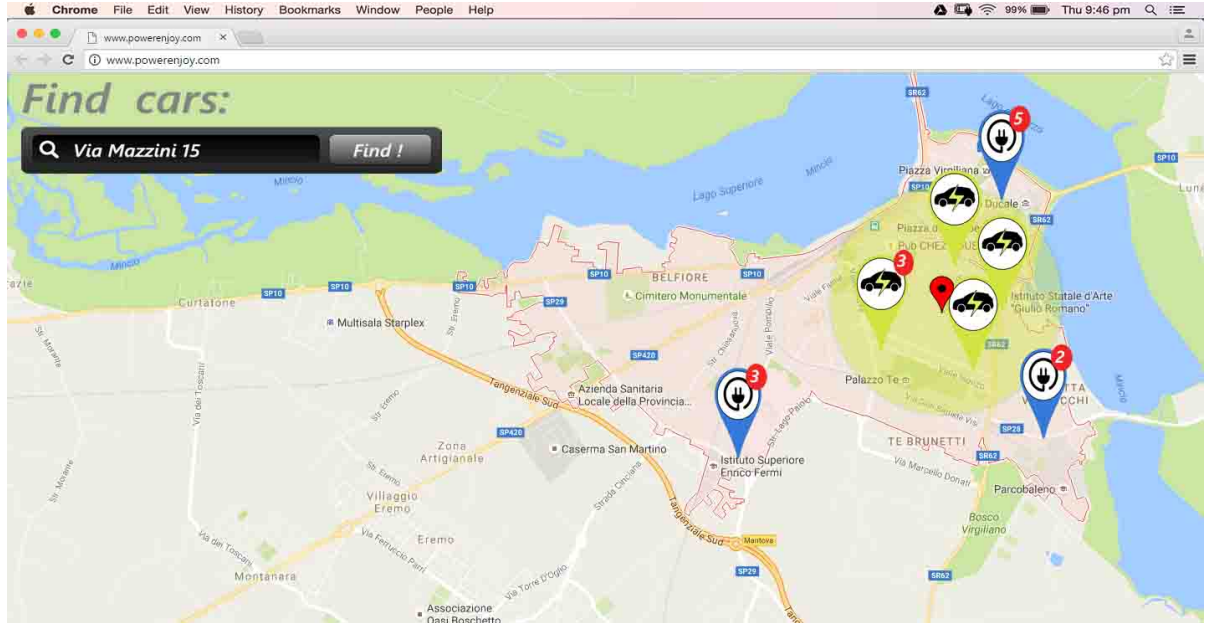


Figure 2: View map on the website.

3.2.3 Registration form

The mockup above shows the registration procedure that a guest has to complete in order to become a client and have access to the PowerEnJoy car sharing.

The registration form is divided into three steps: 1 Personal, 2 Upload Documents, and 3 Payment. The '1 Personal' step is currently active.

Gender
☐ Male ☐ Female

First Name

Last Name

Street **No.**

Postcode **City**

E-Mail Address

Confirm E-Mail Address

Country

Mobile phone number

Preferred City

Date of birth

ID number

Licence Number

Next

Figure 3: Registration form.

3.2.4 Insert pin on the car display

This mockup shows the form in which the client has to insert his pin. After inserting the correct pin the client can start the engine of the car.



Figure 4: Insert pin on the car display.

3.3 The world and the machine

“The World and the Machine” model by M. Jackson and P. Zave is useful to identify the entities in the portion of system that has to be developed (**The Machine**), the entities inside the portion of the real-world affected by the Machine (**The World**), and all the **Shared Phenomena** that are controlled by the world and observed by the machine or controlled by the machine and observed by the world.

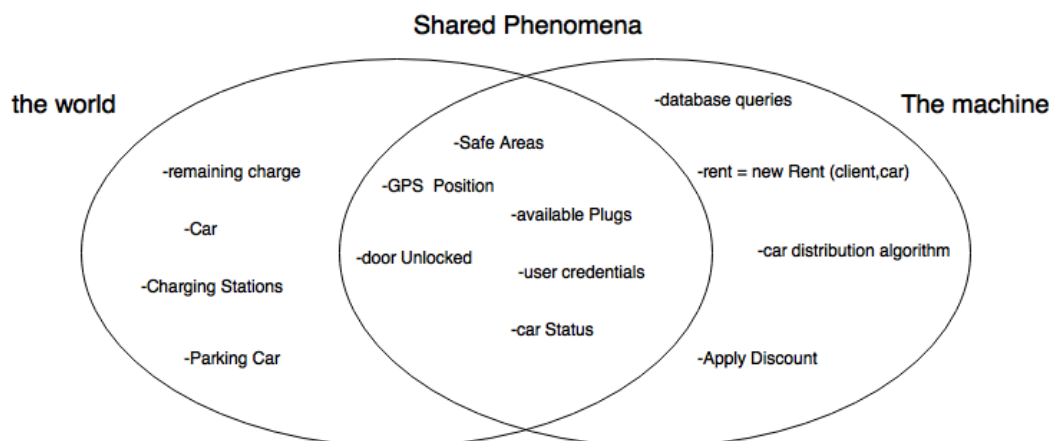


Figure 5: The World and the Machine.

4 Scenarios

4.1 Registration Scenario

Alice heard from his friend Charles that there is a new eco-friendly car sharing service in her town and she wants to use it. She installs the application PowerEnJoy on her device, opens the registration section and fills in the form with her personal data, e-mail and a credit card number. Then, she makes a photo of her driving license and uploads it. The system receives this information and verifies if it is valid by making a request to the national transport authority. So, the system sends back to Alice her password and pin, and allows her to rent a car.

4.2 Log in and display car information Scenario

John has to make a long travel to a venue called 'The wall' and he finds out that the only reasonable way to get there is by using a car sharing service. The only one for which he has registered is PowerEnJoy, but he has never used it before and John does know nothing about that place. In order to plan his route he logs in the web application and sees the available cars, the charging levels, the charging stations near his position and he discovers that 'The wall' is a safe area.

4.3 Reservation Scenario

Bob needs to reach the airport at 3 am but there are no public transportation at that hour and the taxis are too expensive for him. So he decides to use the PowerEnJoy application from his smartphone. Once found the closest available car on the map, he reserves it and starts moving to the car location.

4.4 Opening / Rental scenario

After 10 minutes of walk, Bob is in front of the car, he opens it using the app and he puts the luggage in the trunk. After inserting the pin, he starts the engine and selects the destination. At this point the rent starts and Bob makes his way to the airport. While Bob is driving to the airport, he can check how much he is spending in the car display.

4.5 Ecology Scenario

Charles loves the PowerEnJoy philosophy of respecting nature and he uses the service every day to go work, there is a power station near his office so each time he parks the car there and plugs it into the power grid to get the discount. The office, is located in a safe zone so whenever the power station is full he leaves the car in the nearby and he always checks that the system closes the car before leaving.

4.6 Reporting dirty car Scenario

Hahn is a solo singer and he wants to make a walk in the park with his dog Chew and his child Loren. At the end of the day they are very tired and they don't want to come back home walking so they decide to use PowerEnJoy. The system finds that there is a car nearby, in front of an ice-cream shop. When they arrive to the car Loren cannot resist to the temptation of the ice-cream so he starts crying and Hahn to makes him stops decides to buy him an ice-cream, but in the meanwhile Chew is playing in the mud because he is feeling alone. Hahn does not want to make the car dirty but he sees that Jabba, a friend of him, is coming. Han has to pay Jabba a conspicuous amount of money and since Hahn doesn't have that money yet he decides to jump in the car. Hahn is used to driving very fast and near a traffic light he slams on the brakes and Loren drops the ice-cream, Chew jumps to the front seat and eats it. After arriving at the destination they leave the car all dirty. After a while Anakin reserves the same car but he finds it dirty so he uses the function provided by the application and signals the abuse. PowerEnJoy applies a fee to Hahn and blocks him for two months.

4.7 Saving Mode Scenario

Donnie, Ester, Frank and Gabriela are students and they want to travel in the cheapest possible way. Today is Gabriela's birthday and they want to go to the mall to celebrate and since Gabriela is eco-friendly they decide to go with an electric car. Frank has a PowerEnJoy account so they decide to use the service and since it's cheaper they use the money saving option. The application suggests them the nearest car with enough charge to reach the mall, then it gives information to get to the car and once opened it the display inside the car suggests the charging stations that are nearest to the mall. When they arrive they plug the car in the charging grid. The system recognizes that they plugged the car, they left the car with more than 50% of residual charge, and that there were 4 passengers, so it applies $10\% + 30\% + 20\%$ of discount.

5 UML models

5.1 Use Cases Description

5.1.1 Register to PowerEnJoy

Actors: Guest

Entry Conditions: A guest enters in the registration section.

Flow of events:

1. The guest fills the form with his personal data, the number of his ID card and the number of the driving license.
2. The guest uploads driver license picture.
3. The guest fills the payment information form with his credit or prepaid card's information.
4. PowerEnJoy verifies if the data are valid and sends an email to the guest with his password and a pin necessary to start the rental.
5. PowerEnJoy registers the guest as a new client.

Exit Conditions: Guest successfully ends registration process and becomes a Client. From now on he can log in to the application using his credential and use the PowerEnJoy service.

Exceptions:

- If the guest inserts not valid data the system highlights the not valid fields and doesn't allow him to complete the registration.
- If the guest inserts an already used Email PowerEnJoy doesn't allow him to complete the registration.
- The system does not allow an existing client to register a second time.

Special Requirements:

- After the registration the client must be able to use the service after no more than 5 minutes.
- After that the guest has filled the form, he must receive an email feedback with the password and pin within two hours.
- The communication of sensible data is made throw a protected channel.

5.1.2 Log-in

Actors: Client

Entry Conditions: A client enters in the log-in section.

Flow of events:

1. The client enters his credentials.
2. PowerEnJoy verifies if credentials are valid.
3. PowerEnJoy redirects the client to the map section.

Exit Conditions: Client is logged in to the system.

Exceptions:

- The client inserts not valid credentials. PowerEnJoy notifies him an error and allows him to enter his email and password again.
- If the client is a blocked client the system notifies him an error and doesn't allow him to log-in.

Special Requirements:

- The system should provide a feedback in 10 seconds.

5.1.3 Display a map information

Actors: User

Entry Conditions: Entering in the map section.

Flow of events:

1. User sends his GPS position or selects an address on the map, and specifies a maximum distance from this point.
2. PowerEnJoy displays the data about the part of map requested:
 - location of available cars
 - location of charging areas and number of available charging spots
 - boundaries of safe areas
3. User requests information about one selected car.
4. PowerEnJoy sends back the remaining charge percentage and the license plate number of the selected car .
5. User can see the requested information in a car information menu.

Exit Conditions: Exit from the map section, or a start of a reservation.

Exceptions:

- If User doesn't obtain a valid GPS signal, PowerEnJoy uses a default position like the center of his preferred city.
- If User requests information about a car that is not available anymore, PowerEnJoy doesn't send any information about this car, and all the positions of displayed cars are refreshed.
- If User selects an address out of city, PowerEnJoy uses a default position.

Special Requirements:

- The map will refresh every 10 seconds.

5.1.4 Reserve a car

Actors: Logged Client

Entry Conditions: A client clicks the Reserve button in the car information menu.

Flow of events:

1. PowerEnJoy sets the car as reserved.
2. The client receives the confirmation of the reservation.
3. The client checks the remaining time of the reservation.

Exit Conditions: The car is no longer available, the client has now a rent in progress.

Exceptions:

- The client has already canceled a reservation on the same car in the last 15 minutes.
- The car is no longer available.

In all of these cases the reservation process is not performed and the user is redirected to the map section.

Special Requirements:

- The system should provide a feedback in 10 seconds.

5.1.5 Open a car

Actors: Logged Client

Entry Conditions: Client has done a reservation.

Flow of events:

1. Client moves to a car location, within an hour from the reservation.
2. Once he has arrived, he can push the open button from the application.
3. PowerEnJoy verifies the client position and releases the car doors lock.
4. Client opens the car within a minute.

Exit Conditions: Client starts the rent.

Exceptions:

- If the client doesn't reach the car within one hour, PowerEnJoy deletes the reservation and applies a fee of 1 Euro.
- If the client tries to open the car while he is farther more than 15 meters from the car or he has not GPS signal, PowerEnJoy doesn't unlock the car door.

Special Requirements:

- PowerEnJoy unlocks the car within a minute from the request.

5.1.6 Start the rent

Actors: Logged Client

Entry Conditions: The system has opened the car.

Flow of events:

1. The Client inserts the pin.
2. The system after checking the correctness of the pin enables the possibility to turn on the engine.
3. The Client turns on the engine.
4. The Client selects his destination.
5. The car screen displays the current cost and route information.
6. The Client starts moving to his destination.
7. The Client sees in real time how much he is spending.

Exit Conditions: The Client is driving the rented car.

Exceptions:

- If the charge of the car is lower than 15% the car system signals to park the car immediately.
- If the client continues driving until the car battery is empty the system applies a fee of 100 euros to the client.
- If the rent takes more than 1 day the system signals the case to the operators system.

Special Requirements: There are no special Requirements.

5.1.7 End the rent

Actors: Logged Client

Entry Conditions: Client reaches the destination.

Flow of events:

1. Client parks the car in a safe area and turns off the engine.
2. Client pushes the button to end the rent and exits the car.
3. PowerEnJoy receives the request, checks if the car is parked in a safe area and, after one minute, locks the doors of the car and terminates the rent.
4. PowerEnJoy charges the client after 5 minutes to give him the possibility to plug the car.

Exit Conditions: The car is set as available.

Exceptions:

- If the client tries to leave the car out of a safe area, the system doesn't allow him to end the rent.
- If the client doesn't have enough money to pay the rent, PowerEnJoy disables the client account until the debt is absolved.
- If the client doesn't exit from the car after one minute, the system doesn't lock the car doors and doesn't terminate the rent.

Special Requirements: There are no special Requirements.

5.1.8 Plug the car

Actors: Logged Client

Entry Conditions: The Client parks the car in a charging station with free plugs and ends the rent.

Flow of events:

1. The Client opens the charging slot of the car.
2. The Client plugs the plug to the car.
3. The car system applies a discount of 30% to the last rent.

Exit Conditions: The car is correctly plugged-in and is recharging.

Exceptions: If the client plugs the car after 5 minutes, the system doesn't apply the discount.

Special Requirements: There are no Special Requirements.

5.1.9 Report damaged/dirty car

Actors: Logged Client

Entry Conditions: Client is in front of a damaged or dirty car he has reserved.

Flow of events:

1. A Client decides to report the conditions of the car by clicking the "Report damaged/dirty car" button on the app.
2. PowerEnJoy redirects the client to a form where he has to indicate:
 - Type of Abuse : "Damaged" or "Dirty"
 - Gravity of the problem : from 0 to 5
3. PowerEnJoy signals the case to the operators system.
4. PowerEnJoy proposes to the client two alternatives :
 - "Continue rent"
 - "End rent"

5. The client selects the desired choice.

Exit Conditions: The car is set unavailable.

Exceptions: There are not exceptions.

Special Requirements: There are not Special Requirements

5.1.10 Select money saving option

Actors: Logged Client

Entry Conditions: The Client selects the money saving option on the car display.

Flow of events:

1. The system estimates which is the most suitable charging station based on the city car distribution and the location chosen by the client.
2. The car system suggests a possible charging station.
3. The client selects the suggested charging station.

Exit Conditions: The car system starts the navigation to the chosen charging station.

Exceptions:

- If the client doesn't like the suggested charging station he can undo the action and proceed with the normal mode.

Special Requirements:

- The system should provide feedback within 10 seconds.

5.2 Use Case Diagram

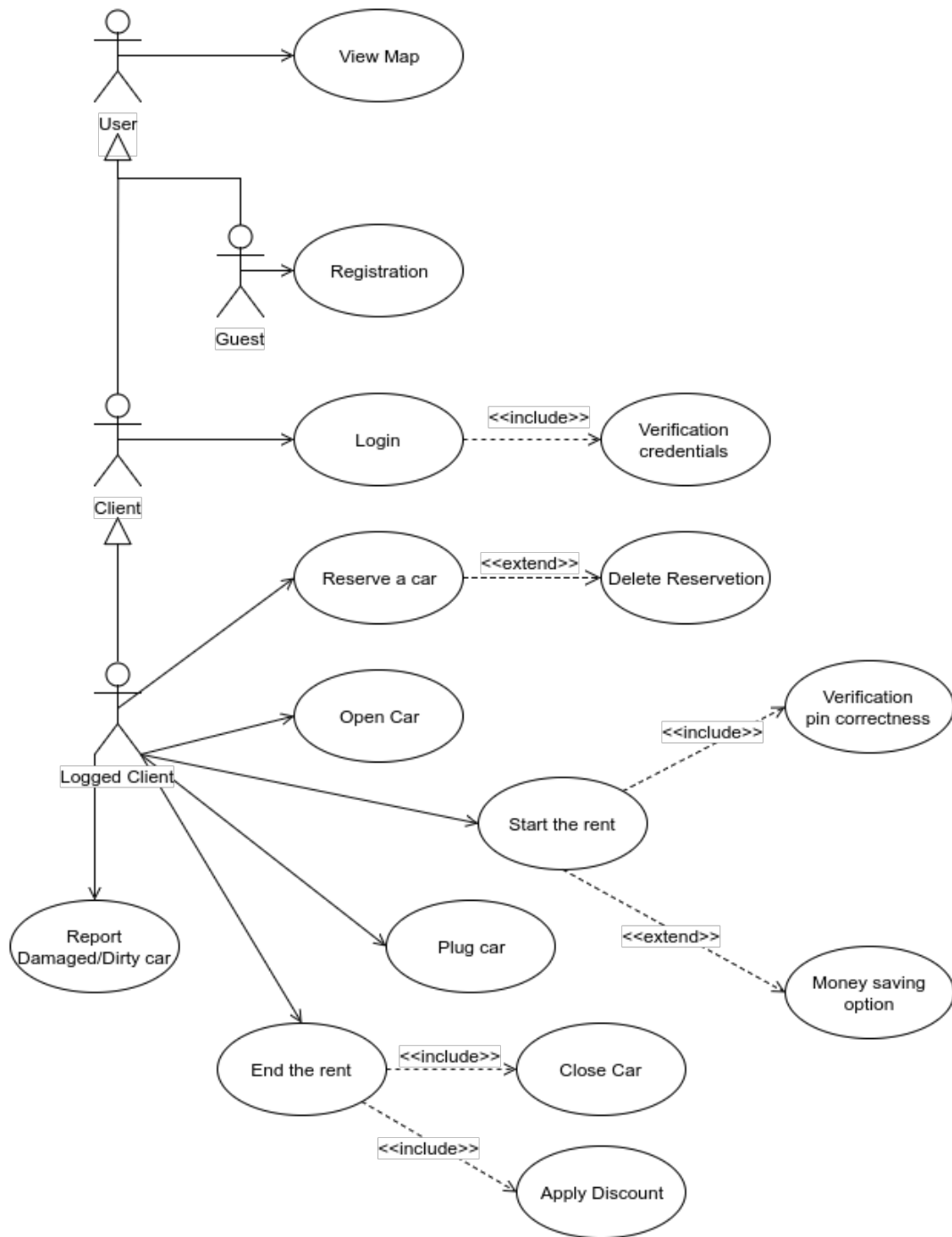


Figure 6: Use Case Diagrams.

5.3 Class Diagram

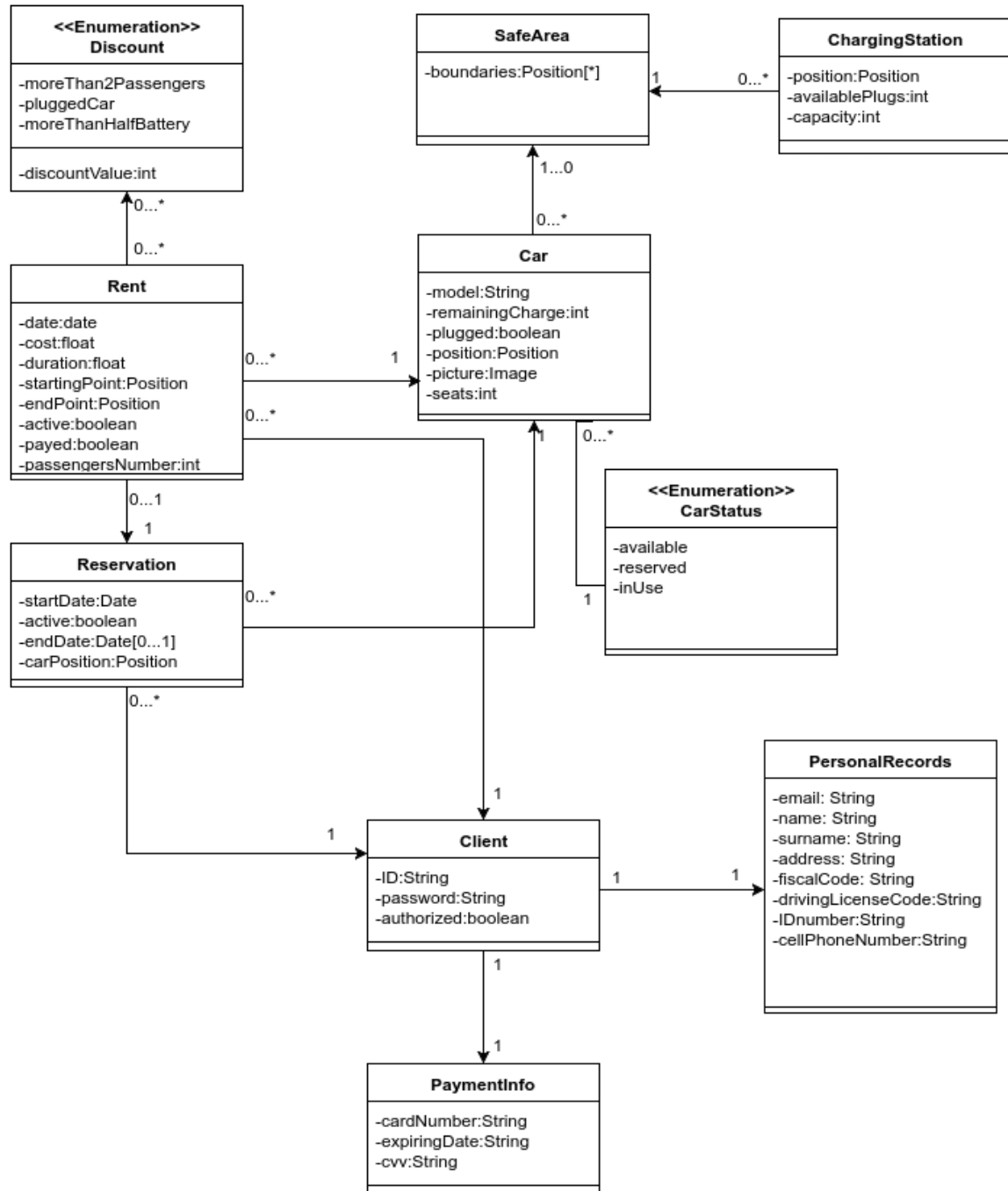


Figure 7: Class Diagrams.

5.4 Sequence Diagrams

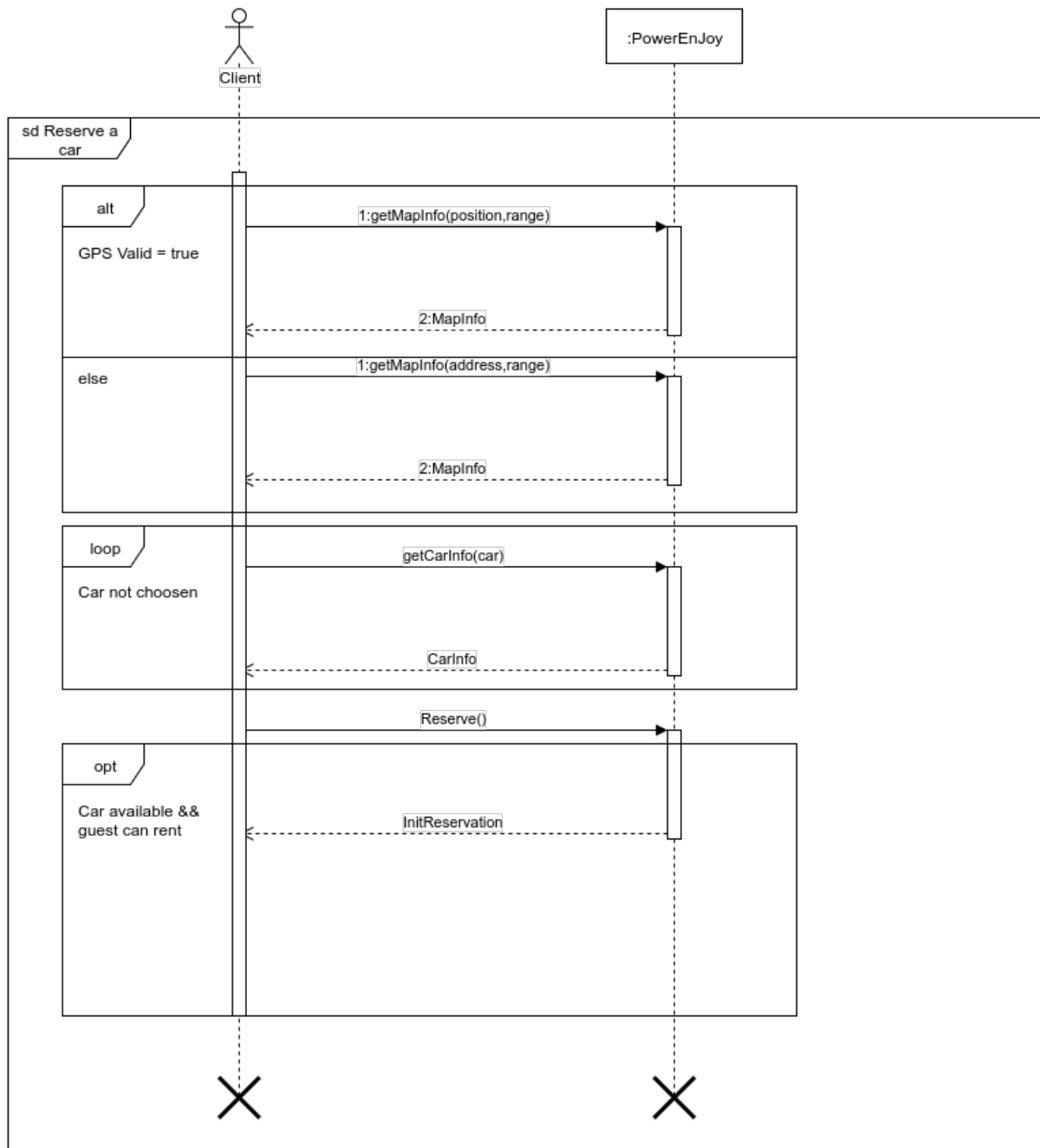


Figure 8: Reserve a car sequence diagram.

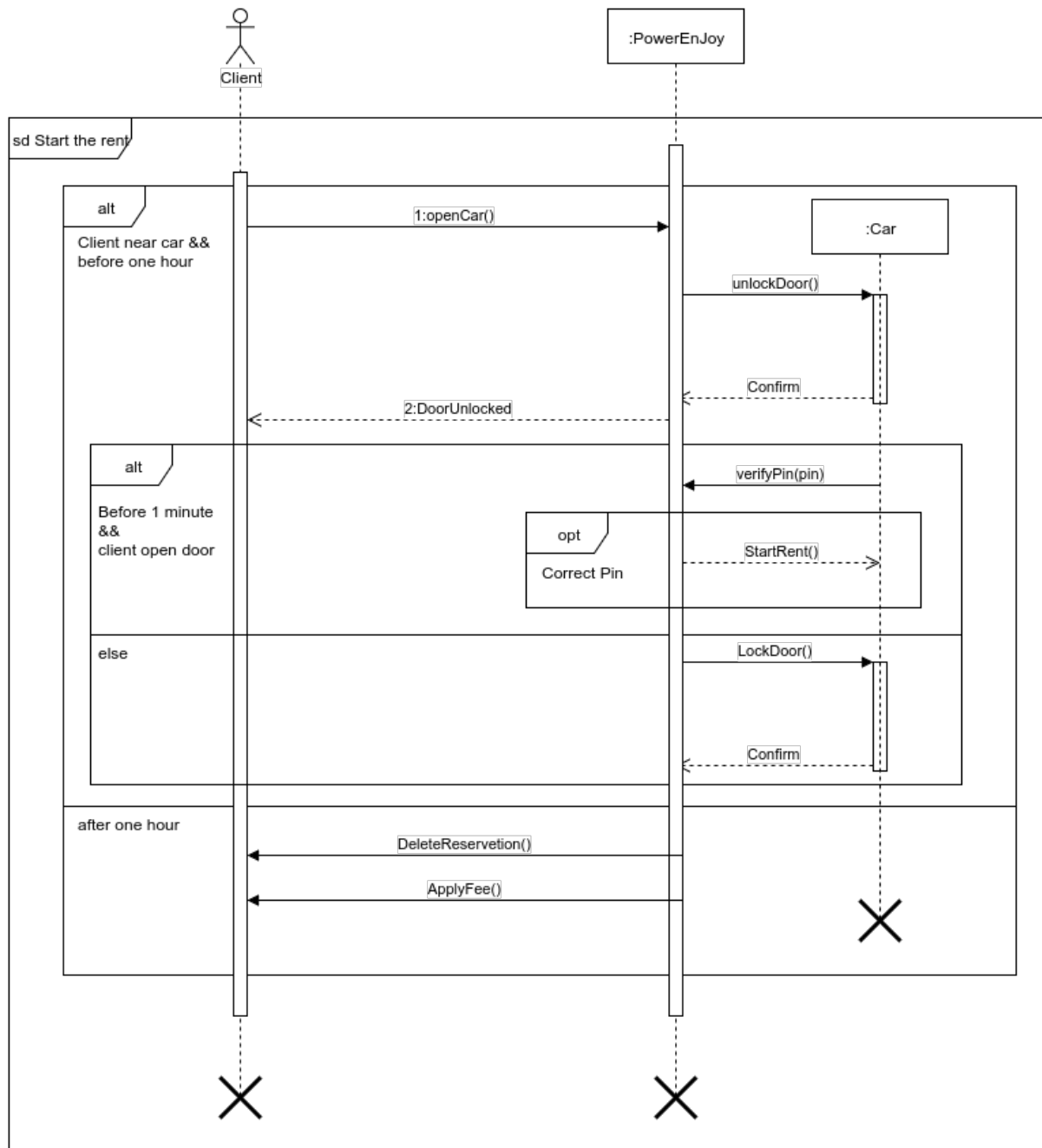


Figure 9: Start the rent sequence diagram.

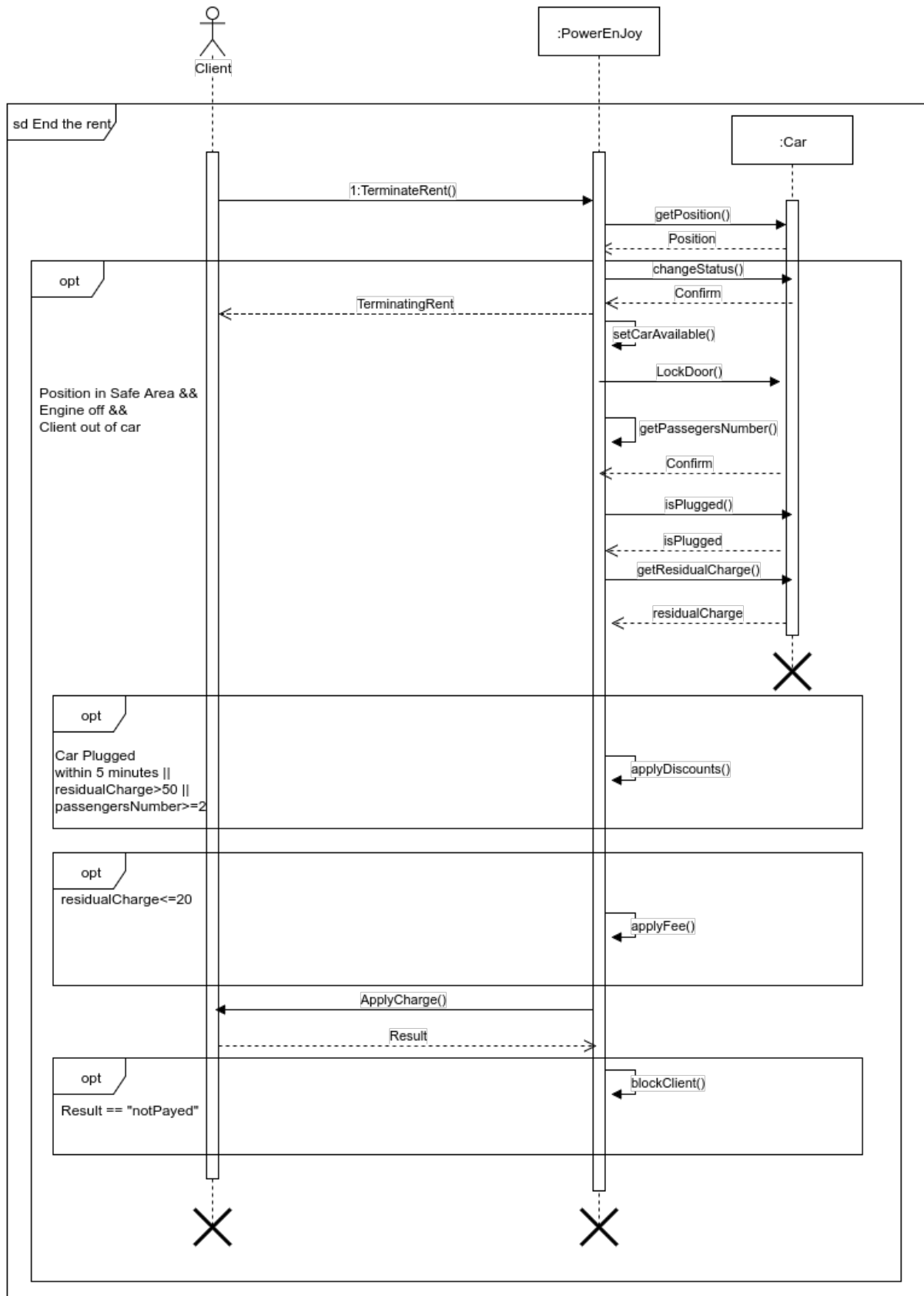


Figure 10: End the rent sequence diagram.

6 Alloy

6.1 Code

```
open util/boolean
```

```
enum CarStatus {  
    available ,  
    reserved ,  
    inUse  
}
```

```
sig Position{}
```

```
sig ChargingStation{  
    capacity : Int ,  
    availablePlugs : Int ,  
    parkedCars : set Car ,  
    position : Position  
}  
{  
    capacity > 0  
    #parkedCars <= capacity  
    availablePlugs = capacity - #parkedCars  
}
```

```
sig ActiveRents{  
    rents : set Rent  
}
```

```
sig Rent {  
    reservation : Reservation ,  
    client : Client ,  
    car : Car  
}  
{  
    reservation.client=client  
    reservation.car = car  
}
```

```
sig Reservation{  
    client : Client ,  
    car : Car  
}
```

```
sig Car {  
    status : CarStatus ,  
    remainingCharge : Int ,  
    plugged : Bool ,  
    position : Position ,
```

```

}
{
    remainingCharge >= 0
    remainingCharge <= 100
    status = inUse implies plugged = False
}

sig Client {}

//all rents are in Active Rents
fact allRentAreInActiveRents{
    all rent:Rent | rent in ActiveRents.rents
}

//a client cannot have more than one active rent
fact noRentMadeByTheSameClient{
    all r1,r2 : Rent | r1 != r2 implies r1.client != r2.client
}

//for each rent there is a reservation and it's unique
fact noRentWithTheSameReservation{
    all r1,r2:Rent | r1!=r2 implies r1.reservation != r2.reservation
}

//a car cannot be rented by 2 clients or more
fact noRentOnTheSameCar{
    all r1,r2:Rent | r1!=r2 implies r1.car != r2.car
}

//a car cannot have more than one active reservation
fact noReservationOnTheSameCar{
    all r1,r2:Reservation | r1!=r2 implies r1.car != r2.car
}

//a client cannot reserve more than one car
fact noReservationByTheSameClient{
    all r1,r2 : Reservation | r1 != r2 implies
        r1.client != r2.client
}

//all cars parked inside charging stations are plugged
fact {
    all c:Car | c in ChargingStation.parkedCars implies
        c.plugged = True
}

```

```

//cars parked in the charging station have
//the same position of the charging station
fact chargingCarsInTheSameStation{
    all c:Car , cs :ChargingStation | c in cs.parkedCars implies
        c.position = cs.position
}

//if a car is not rented or reserved its state is available
fact availableCarCondition{
    all c:Car | not (c in Rent.car or c in Reservation.car) implies
        c.status=available
}

//if a car is not rented but is reserved its state is reserved
fact reservedCarCondition{
    all c:Car | c in Reservation.car and not ( c in Rent.car ) implies
        c.status = reserved
}

//if a car is rented his state is inUse
fact inUseCarCondition{
    all c:Car | c in Rent.car implies c.status = inUse
}

//Each charging station is located in a different position
fact noChargingStationsInTheSamePlace{
    all cs1,cs2: ChargingStation | cs1 != cs2 implies
        cs1.position!=cs2.position
}

//predicate to end a rent
pred endRent(activeRents ,activeRents' :ActiveRents , rent:Rent ){
    activeRents'.rents = activeRents.rents-rent
}

//predicate to init a rent
pred initRent(activeRents , activeRents':ActiveRents , rent:Rent ){
    activeRents'.rents = activeRents.rents+rent
}

pred show{

}

//check that there are no more charging car than car in that position

```

```

assert noMoreChargingThanParked {
  all cs : ChargingStation , p : Position | p = cs.position implies
    #cs.parkedCars <= #{c : Car | c.position = p}
}

//check that a client has not more than one rent active
assert atMostOneRentPerClient{
  all c : Client | lone rent : Rent | rent.client = c
}

//check that there is no more than an active rent for each car
assert atMostOneRentPerCar{
  all c : Car | lone rent : Rent | rent.car = c
}

//check that there is no more than an
//active reservation for each client
assert atMostOneReservationPerClient{
  all c : Client | lone res : Reservation | res.client = c
}

//check there is no more than one active reservation for each car
assert atMostOneReservationPerCar{
  all c : Car | lone res : Reservation | res.car = c
}

//check the conditions that must be satisfied after an end of a rent
assert conditionEndedRent{
  all activeRents , activeRents' : ActiveRents , rent : Rent , c1 : Car |
    endRent[activeRents , activeRents' , rent] and c1 = rent.car implies
      not c1 in activeRents'.rents.car
}

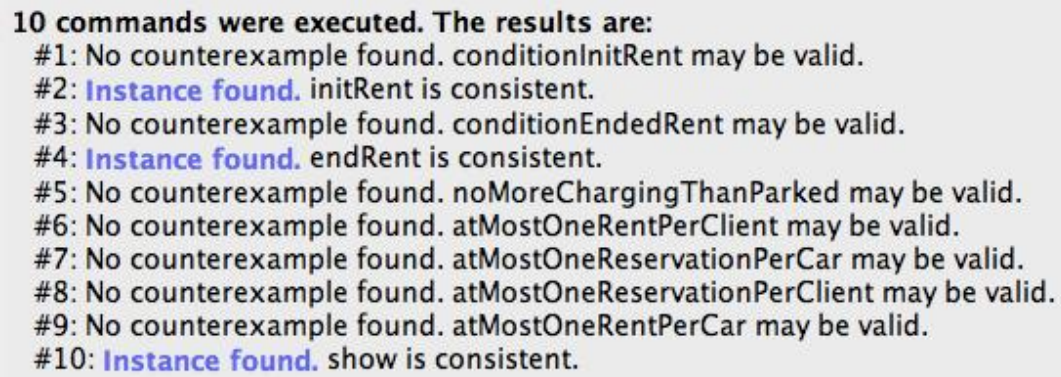
//check the conditions that must be satisfied after a start of a rent
assert conditionInitRent{
  all activeRents , activeRents' : ActiveRents , rent : Rent , c1 : Car |
    initRent[activeRents , activeRents' , rent] and c1 = rent.car implies
      c1 in activeRents'.rents.car
}

check conditionInitRent for 3
run initRent for 5 but exactly 2 Rent
check conditionEndedRent for 5
run endRent for 5
check noMoreChargingThanParked for 3
check atMostOneRentPerClient for 5
check atMostOneReservationPerCar for 5
check atMostOneReservationPerClient for 5
check atMostOneRentPerCar for 5
run show for 5

```

6.2 Results

This screenshot of the Alloy Analyzer software that shows the consistence of the model in all parts.



10 commands were executed. The results are:

- #1: No counterexample found. conditionInitRent may be valid.
- #2: **Instance found.** initRent is consistent.
- #3: No counterexample found. conditionEndedRent may be valid.
- #4: **Instance found.** endRent is consistent.
- #5: No counterexample found. noMoreChargingThanParked may be valid.
- #6: No counterexample found. atMostOneRentPerClient may be valid.
- #7: No counterexample found. atMostOneReservationPerCar may be valid.
- #8: No counterexample found. atMostOneReservationPerClient may be valid.
- #9: No counterexample found. atMostOneRentPerCar may be valid.
- #10: **Instance found.** show is consistent.

Figure 11: Alloy results.

6.3 Generated world

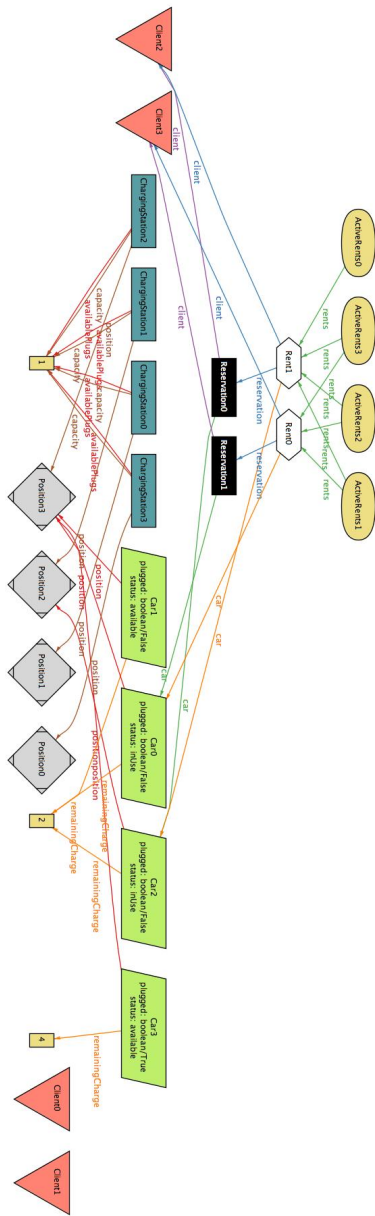


Figure 12: The world generated.

7 Appendix

7.1 Software and tool used

The tools we used to create this RASD document are:

- Overleaf (for latex writing in parallel)
- Photoshop (for mockups)
- Draw.io (for UML diagrams)
- Alloy

7.2 Hours of work

During the whole project the team worked together each time with the following schedule:

Claudio Salvatore Arcidiacono

- 31 October : 16-20 4hours
- 1 November : 15-21 6hours
- 4 November : 14-21 7hours
- 5 November : 16-20 4hours
- 6 November : 14-20 6hours
- 7 November : 17-22 5hours
- 9 November : 18-21 3hours
- 10 November : 18-21 3hours
- 11 November : 11-22 (1hour of pause) 10hours
- 12 November : 10-24 12hours (2hours of pause)
- 13 November : 10-18 6 hours (2hours of pause)

Total hours: 63 hours.

Antonio Di Bello

- 31 October : 16-20 4hours
- 1 November : 15-21 6hours
- 4 November : 14-21 7hours
- 5 November : 16-20 4hours
- 6 November : 14-20 6hours
- 7 November : 17-22 5hours
- 9 November : 18-21 3hours
- 10 November : 18-21 3hours
- 11 November : 11-22 (1hour of pause) 10hours
- 12 November : 10-24 12hours (2hours of pause)
- 13 November : 10-18 6 hours (2hours of pause)

Total hours: 63 hours.

Denis Dushi

- 31 October : 16-20 4hours
- 1 November : 15-21 6hours
- 4 November : 14-21 7hours
- 5 November : 16-20 4hours
- 6 November : 14-20 6hours
- 7 November : 17-22 5hours
- 9 November : 18-21 3hours
- 10 November : 18-21 3hours
- 11 November : 11-22 (1hour of pause) 10hours
- 12 November : 10-24 12hours (2hours of pause)
- 13 November : 10-18 6 hours (2hours of pause)

Total hours: 63 hours.

7.3 Changelog