



PowerEnJoy

Software Engineering 2 Course Project

Requirement Analysis and Specification Document

Niccolo' Raspa
Matteo Marinelli

Table of Content

1. **Introduction**

- 1.1. Purpose
- 1.2. Scope
- 1.3. Goals
- 1.4. Definitions, acronyms, and abbreviations
- 1.5. Text Assumptions and Policies
- 1.6. Overview

2. **Overall Description**

- 2.1. Domain Properties
- 2.2. Actors
- 2.3. Electric Car
- 2.4. Scenarios
- 2.5. Use Cases
- 2.6. Class Diagram

3. **Functional Requirements**

4. **Non-Functional Requirements**

5. **Alloy**

- 5.1. Code of the Model
- 5.2. Asserts
- 5.3. World Generated

1 Introduction

1.1 Purpose

The purpose of this document is to describe all the requirements and the specifications for the PowerEnjoy System.

It specifies goals of the software to be and all the functionalities expected.

We will also list all the non-functional requirements and assumptions we've made. We have included possible scenarios, use-cases and UML diagrams to help clarify the scope of the system and gain a deeper understanding of the software to be.

This document is intended to be used by the members of the project team that will implement and verify the correct functioning of the system.

1.2 Scope

PowerEnjoy is a digital management system for a car-sharing service that exclusively employs electric cars. It allows registered clients (Power Users) to use a vehicle paying only on the basis of the actual use during each individual rental. In addition to the classic functionalities expected on a car sharing service, it will also incentivize environment-friendly and virtuous behaviours of the Power Users, providing discounts on the regular ride fee.

PowerEnjoy employs a single model of electric car which comes with pre-installed sensors that can be used to monitor the car status (battery level, location, etc..) at any given time.

The car can be picked up from a safe areas or from a Power Enjoy Charging Stations, and should be left in another safe area.

The Power User will be charged only for the time he has rented the car, there are no subscriptions or additional prices to use the service.

Our system interacts with two external services:

- A Payment Service
- An Assistance Service

The role of this two systems and the interaction will be cover in the next chapter.

1.3 Goals

1. Allows USER to register providing credentials and payment informations
2. Allows USER to login to the system
2. Allows POWER USER to modify its personal informations
4. Allows POWER USER to see locations and battery levels of available CARs within a specific distance from a location (eg. POWER USER location or specified address)
5. Allows POWER USER to reserve one AVAILABLE CAR in a SAFE AREA.
6. Allows POWER USER to cancel a CAR RESERVATION
7. Allows POWER USER to unlock his RESERVED CAR when he is nearby (less than 5 meters)
8. CAR RESERVATION expires after one hour
9. CAR RESERVATION expiration causes 1€ charge to the POWER USER
10. Allows POWER USER to know the current RIDE FEE at any time through a screen in the RESERVED CAR
11. RIDE FEE will be calculated with per-minute price from the moment of the engine ignition until the moment in which the POWER USER leaves the car
12. The FINAL FEE will be calculated applying discounts/fines to the RIDE FEE according to the power enjoy policy
13. POWER USER will be charged of the FINAL FEE after he exits the RESERVED CAR.
14. CARs are automatically locked when parked and the POWER USERS gets out
15. POWER USER with pending payments can't reserve cars

1.4 Definitions, acronyms, and abbreviations

- USER

A person who is not authenticated to the system.

- POWER USER

Main user of the application, he's a registered user who can can reserve cars.

- RENT

Interval of time in which a car can be used from a Power User which has previously reserved it.

- RESERVATION

Interval of time necessary for the Power User to reach and ignite his selected car. It can last up to 1 hour.

- RIDE

Time interval from the ignition of the motor until the driver and all the passengers exit the vehicle.

- SHARED RIDE

Percentage of RIDE with at least 2 other passengers on board.

- PASSENGER

A person inside the car other than the driver, not necessarily registered to the system.

- CAR / ELECTRIC CAR

A specific model of an electric car which can be rented from Power Users.

- CAR DISPLAY

It's a display pre-installed inside the car, where the user can see the RIDE FEE.

- AVAILABLE CAR

A car parked in a safe area which can be rented.

- RESERVED CAR

It's a car currently unavailable because a Power User made a reservation.

- RENTED CAR

It's a car currently unavailable because a Power User has either reserved it or he's driving it.

- MALFUNCTIONING CAR

A car that fails to function normally and requires assistance.

- BATTERY

Refers to the residual battery level of a Car.

- CHARGING STATION

It's a station owned by Power Enjoy where cars can be re-charged and picked up from Power Users.

- SAFE AREA

A predefined set of parking areas where we can assume a car can be safely parked without getting fines.

- ASSISTANCE SERVICE / AS

An external service that provides assistance to cars in case of malfunctioning. It also charges cars on site and brings cars from unsafe areas to safe areas.

- PAYMENT SERVICE / PS

An external service that handles payments from Power Users.

- RIDE FEE

It's a variable amount of money proportional to the length of the Ride.

- FINAL FEE

It's the amount of money that the Power User is charged after the rent is over. It includes discounts and fines.

1.5 Text Assumptions and Policies

1.5.1 Text Assumptions

The text doesn't specify what should happen if a user parks in a unsafe area. Since it's really restrictive to prevent this situation from happening we've decided the following:

- As soon as a Power Uses turns the engine off, it will be notified through the car display if he has parked in an unsafe area.
- If he decide nevertheless to exit the vehicle, the rent will not stop and the user will continue to be charged.
- The Power User can still unlock the car and park it a safe area.
- If the car is left in the same unsafe area for more than 1 hour, the rent will eventually end.

- The car will become unavailable.
 - The AS will be notified to take the car back to a safe area.
 - User will be charged for the extra time to cover the costs.
 - Fines may still apply.
- The text doesn't specify how payments should be handled, we supposed there is an external payment service (e.g. Paypal) that takes care of processing payments.

1.5.2 Discount Policy

As said earlier Power Enjoy not only allows to use a vehicle paying only on the basis of the actual use during each individual rental but also incentivizes virtuous behaviour providing discounts to the regular ride fee.

The discounts offered are the following:

1. If a Power User takes at least two other passengers onto the car, the system applies a discount of 10% for the shared ride.
2. If a car is left with more than 50% of the battery, the system applies a discount of 20% on the last ride.
3. If a car is left at Power Enjoy Charging Station 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.

Discounts are only applied if a car is parked in a safe area.

To qualify for a Shared Ride discount a Power User must take 2 passengers for at least the 90% of the Ride length.

Discounts are cumulative except for discounts 2. and 3.

This means that if a user recharges a car with more than 50% of residual battery, the final discount applied is 30%.

This is to avoid early recharges and guarantee a better distribution of the cars.

To provide a fair and equal car sharing among all customers, Power Enjoy penalizes unfair usage of the service:

4. If a car is left at more than 3 KM from the nearest Power Enjoy Charging Station or with less than 20%, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.

If a ride qualifies for both a discount and a fine, only the fine will be applied.

1.5.3 Legal Policy

The system must require his users the permission to get his position and to manage sensible data respecting the privacy law.

To provide financial protection against physical damage and/or bodily injury resulting from traffic collisions and against any liability every Power Enjoy Vehicle has stipulate a full insurance.

To avoid legal problems when registering the user acknowledges that to be covered by the insurance the driver must always be the same person whose account has been used to reserve the car. In case this is not true and for all the cases not covered by our insurance the Power User will take full responsibility of any third party actions.

In other words, Power Enjoy doesn't prevent a Power User from delegating some else as the driver but only the Power User would be held responsible for any possible damage caused.

In any case, the following conditions always hold:

- Any driver must be in possession of a valid driving license
- The vehicle is to be driven in full compliance with the Highway Code, the Civil Code, the Penal Code and in general with maximum diligence
- Any financial penalties for violation of the legal rules relating to the movement of vehicles will be notified to the Power User who undertakes to pay the fines
- Vehicles must not be stolen or parked in private areas

1.5 Overview

The remaining part of the document is organized as follows:

Chapter 2 provides an overall description of the system, analyzing both the surroundings of the system and the domain in which is implemented. We provide an high level view of the system to be and a summary of its expected functionalities.

Chapter 3 focus on the functional requirements. We list them following the order of the use cases introduced in the previous chapter. We will also provide a traceability matrix to emphasize the correctness of our requirements and to ease the trace of the deriving process.

Chapter 4 will discuss the non-functional requirements and their impact on the overall quality of the system.

Chapter 5 will provide a formal specification of the system in Alloy. We will test the correctness of our model and we present the world we've generated.

2 Overall description

This section will provide an overall description of the whole system.

First, we start by describing the environment and its surrounding: we list all the domain properties and the external services we need to interact with.

In the second part we start identifying actors and describe simple scenarios. Finally, we generalize providing both the usecase diagram and the class diagram.

2.1 Domain Properties

In our analysis we assumed the following properties:

1. GPS always gives the correct position
2. Every car has an always available GPS.
3. Cars are able to determine the number of passengers currently on board.
4. Cars are able to detect current battery level.
5. Cars are always able to detect and signal to the system if they are malfunctioning.
6. Every car is equipped with an onboard display.
7. Cars with locked door cannot be opened from the outside
8. System knows the car status at any time.
9. The car notifies the system when the engine has been ignited/turned off.
10. The system has remote control of the cars.
11. Every power user can be geolocalized.
13. The set of safe areas and power grid stations is predefined
15. The system knows when a car has been plugged for recharge
16. The system is able to verify user and payment credentials

2.1.2 External Services

As we said earlier our system interacts with two external services.

In this section we describe such system in details and their expected functionalities.

Payment Service PS

Every Power Enjoy user, in order to use our service, is required to have an account registered to a third party payment service, with a valid payment method.

Power Enjoy will calculate for any renting the final fee but the actual payment will be handled by the payment service (PS).

The PS will be notified of the amount of money owed from which Power User and we will be informed when such payment occurs.

A user with a pending payment will not be able to reserve any car.

PS Assumptions

- 17. There exists an third party payment service for processing payments
- 18. Every user is registered with a valid payment method to the external payment service.
- 19. Payment service notifies as soon as a payment occur

Assistance Service AS

Power Enjoy is in charge of the management of the car-sharing system. All the secondary functionalities are handled by an assistance service, which is responsible for:

- Ordinary Maintenance of the vehicles
- Cleaning of the vehicles
- Repairing malfunction vehicles
- Recharging vehicles onsite
- Take care back from unsafe area to safe areas

AS Assumptions

- 18. AS automatically takes care of fixing malfunctioning cars
- 19. AS automatically takes care of recharging of site cars with less than 20% of Battery
- 19. AS automatically dispatches employees to bring cars from unsafe areas back to safe areas
- 20. AS notifies when car are available again providing, eventually, new location and battery status

2.2 Actors

Our system has to interact mainly with 5 actors.

- USER
- POWER USER
- CAR
- AS
- PS

2.3 Electric Car (Hardware Constrains)

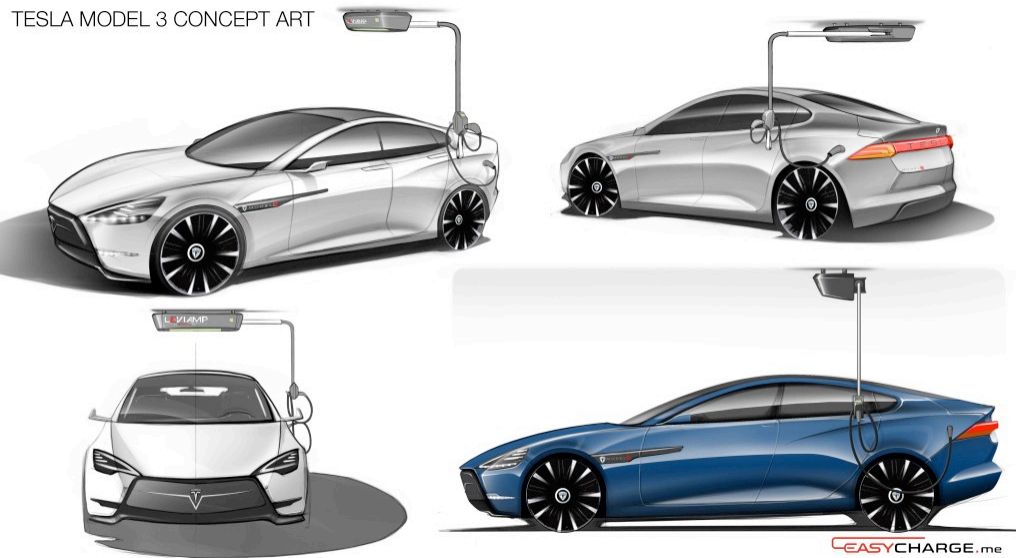
PowerEnjoy service employs a single model of electric car in the first release, but the system is designed to allow any electric vehicle as long as it provides the minimum functionality needed:

- **Weight sensors** to detect the number of people inside
- **Ignition sensors**
- **Battery Level sensors**
- Global Positioning System (GPS)
- Automatic keyless entry
- Remote control
- Lcd touchscreen
- Internet connectivity

Other models should also consider this non functional requirements that will highly affect the quality of the service we provide:





- Battery Length
- Charging Time
- Safety
- Comfort
- Performance
- Navigation System

The car of our system is a Tesla Model 3.



It was provided by Flavio, and it has:

- 5 Seats
- 215 miles Range
- Fast Charging
- High Safety

 <p>Designed to achieve 5-Star Safety Rating</p>	 <p>215 miles Range per charge</p>	 <p>Seating for 5 Adults</p>	 <p>Supercharging</p>
--	--	--	--

The car also has a Display which shows to the user important informations:

- Length of the ride
- Current cost of the ride (before discounts)
- Battery Level

It will (optionally) show:

- Map
- Location of the charging stations
- Set of safe areas



The car itself provides some handy functionalities:

- The car automatically notifies the user when the battery is low
- The car automatically notifies the user when he/she parks on an unsafe area
- The car automatically sends to the server constant updates on his status
- The car automatically notifies both the system and the user when a malfunction occurs

Car State and State Diagram

The Car during his lifetime evolves following these states:

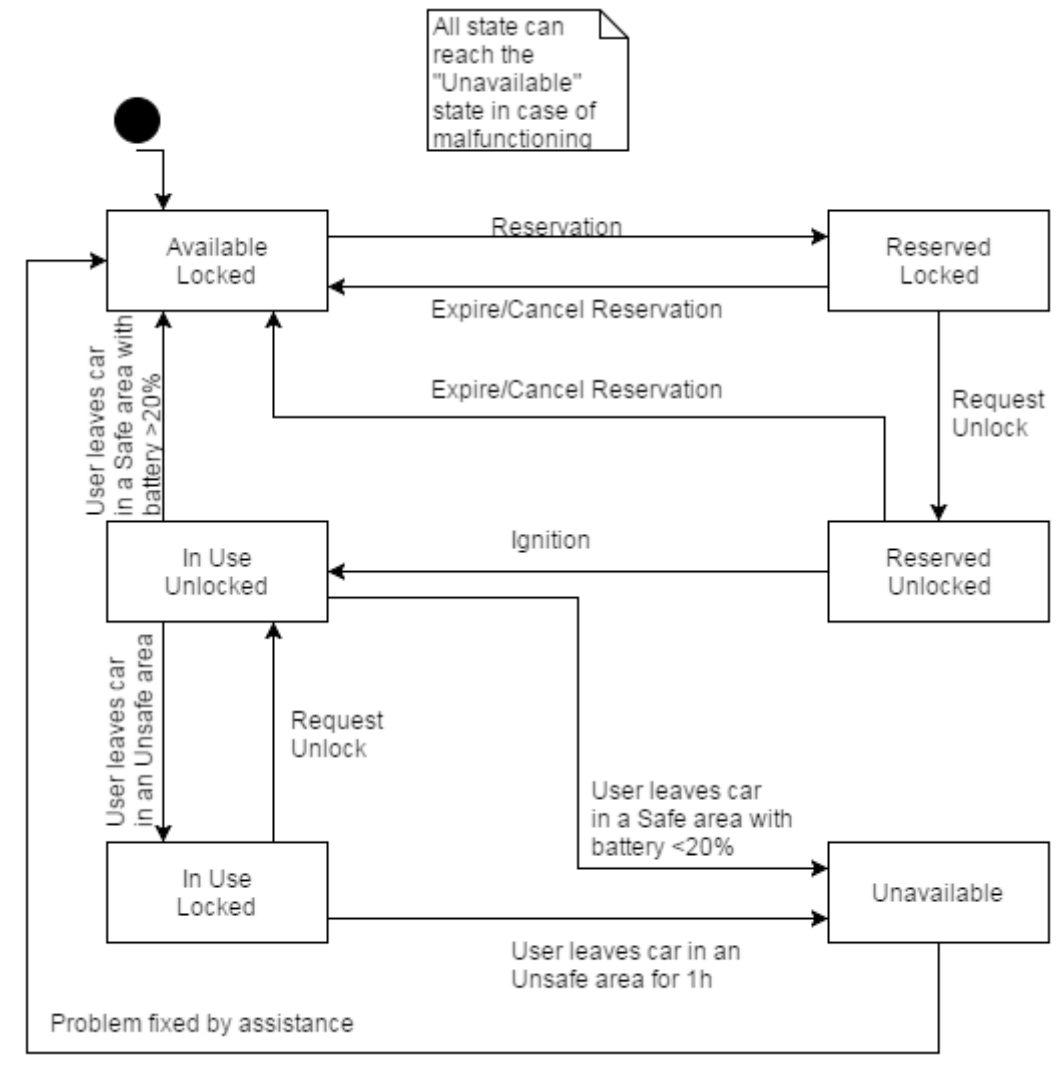
- **Available:** a car can be reserved from a Power User.
- **Reserved:** is a car selected by a Power User to exploit the service.
- **InUse:** is car ignited at least one time after a reservation.
- **Unavailable:** is a car who needs assistance.

There are few additional states:

lock: a car with car doors locked which prevents people to access it

unlock: a car with car doors unlocked

Charging: a car currently charging on a Charging Station. A car can not be charging if it is “InUse” or “Reserved and Unlock”. The car, in fact, deny the unlock if it is linked to a plug.



2.3 Scenarios

In this section we provide some common scenarios that may occur during an instance of the usage of the system.

It will provide an informal but yet real feel of the system to be.

Scenario 1: Registration

Is Friday morning. Alice want to try the Power Enjoy service, she takes hers smartphone and downloads the app. To use the app is needed a registration. In the registration process, Alice need to insert hers personal informations (name, surname, driving licence number...) and a payment service. Alice fills all the fields with valid informations. Alice receive a mail that confirms the registration and the password. Another mail is stent to explain all the correct behaviours to use Power Enjoy system.

Bob want to use Power Enjoy as well, but during the registration makes a mistake inserting the driving license number. Bob try to submit the form but the system recognize the mistake. The problem is notified to Bob and a new form is send.

Scenario 2: Discount Policies

Carl is a Power Enjoy registered user. Carl is late for the Software Engineering 2 cass, because at 8:30 in the morning is quite hard to be in time, and decide to use a Power Enjoy car to reach the university quickly. Carl insert his username and his password correctly. Carl decide to search a car in a range of 150m from his position. Carl select the nearest car and reach the university. At the end of the rent, the is in a safe area with 18% battery. According to the discount policies, 30% of the total fee is added to the final fee that is send to Carl.

The following day Carl faces the same situation, but with him there are two friends. The selected car is leaved in the same conditions, so the 30% of the total fee is added to the final fee, but 10% of the total fee is subtracted to the final fee according to the discount policies.

Scenario 3: Edit Profile

Donald want to modify his Power Enjoy profile. Donald execute the login and select the modifying option. A form equal to the registration form is sent to him. Donald fill all the fields correctly and submit the form. A mail is sent to Donald to confirm the new dates.

Elisabeth need to modify hers profile too. Inserting the informations, Elisabeth write incorrectly her new mail address. After the submission, the realize the mail is not valid, so it will not change any information about Elisabeth. The form is send again t Elisabeth with the notification of the problem.

Scenario 4: Normal Utilization

Flavio is a business man. His hyper luxurious car break down without any reason. Flavio choose to use Power Enjoy, service he financed, to reach his destination. Flavio execute the login and search a car in the range of 278m from his position. Flavio select a car and reach it in 20 minutes. Flavio unlock successfully the car because is exactly near it, so the ride can start. Flavio knows very well the discount policies, so park the car with 49% battery in a safe area not so far from a Power Enjoy grid station, unfortunately full. No discounts are applied and the final fee is sent to Flavio.

George had already selected a car. The selected car is quite far from him, so George can't reach it in one hour. When he realize he will not arrive in time, George try to cancel the reservation, but his mobile lost the connection and the time expire with the reservation still active. The system cancel the reservation and charge George of a fee of 1€.

Scenario 6: Open and Not Switch On.

Harry have to reach the Leaky Cauldron to meet his friends. Harry reserve a car ad reach it quickly. Harry open the car when he is less than 5m far from it and enter inside. Before he can switch on the car, his friend Ronald tells him that the meeting is cancelled. Harry send a curse on his friends and leave the car. The hour is not expired yet, so Harry cancel the reservation and no fee will be charged to his payment service.

Ronald is basically in the same situation of Harry, but he forgot to cancel the reservation so, after one hour, the system will cancel the it by himself and charge Ronald of 1€ fee.

Scenario 7: Car Run Out of Battery

Ines is driving a Power Enjoy car. Ines, even if the car show its battery level, doesn't realize the car is running out of battery, and the battery level reach the zero. The car inevitably stops in an unsafe area. According to the Power Enjoy policies, if a car is parked in an unsafe area, the reservation will continue for an hour, and this will happen to the Ines reservation. After one hour the system will close the reservation and ask to the assistance service to fix the problematic car.

Louis is driving a Power Enjoy car. Like Ines, he doesn't realize the battery level is really low, so the car run out of battery. Louis has great drive skills so is able to bring the car in a safe area before the wheels stops definitively. Since the car is stopped in a safe area, the reservation will end normally. The system will call the assistance to recharge the car.

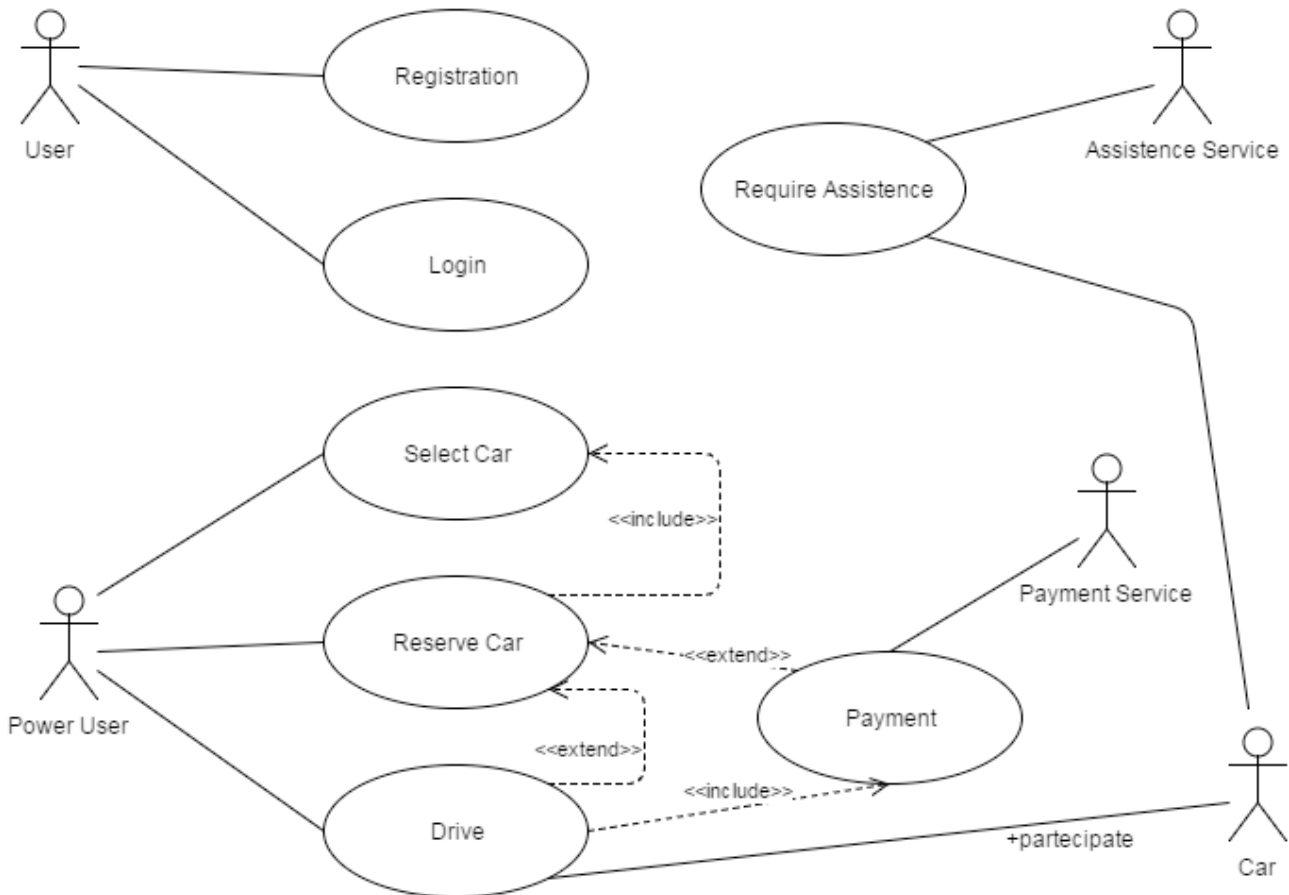
Scenario 8: Park in a Non Safe Area

Matteo is driving a Power Enjoy car. He is not able to find a free place in a save area so park his car in a non safe area, conscious that he will pay an additional hour. The rent of the car is still ongoing, but the system ability Matteo to unlock the car again. After he leaved the car, Matteo see a free place in a surely Safe Area, so he unlock the car and park in a safe area. When he leaves the car, the rent will end normally.

Niccolò is with two friends on a Power Enjoy car. Niccolò is the Power User who rented the car. Niccolò park the car in a non safe area because he is out of Milan. Even if he has two passengers, a 10% will not be applied because the policy don't allows discounts in non safe areas.

2.4 Use Case Diagram

In this section we try to generalize from the scenarios before and list all the use cases. In the next chapter we will follow the order of the use cases to derive all the necessary requirement for the system.



2.5 Class Diagram

In this last section we use a class diagram to give a static global view of the whole system and the domain modeling we implemented.

We choose a UML Class Diagram with an Entity-Control-Boundary Pattern (ECB).

Where:

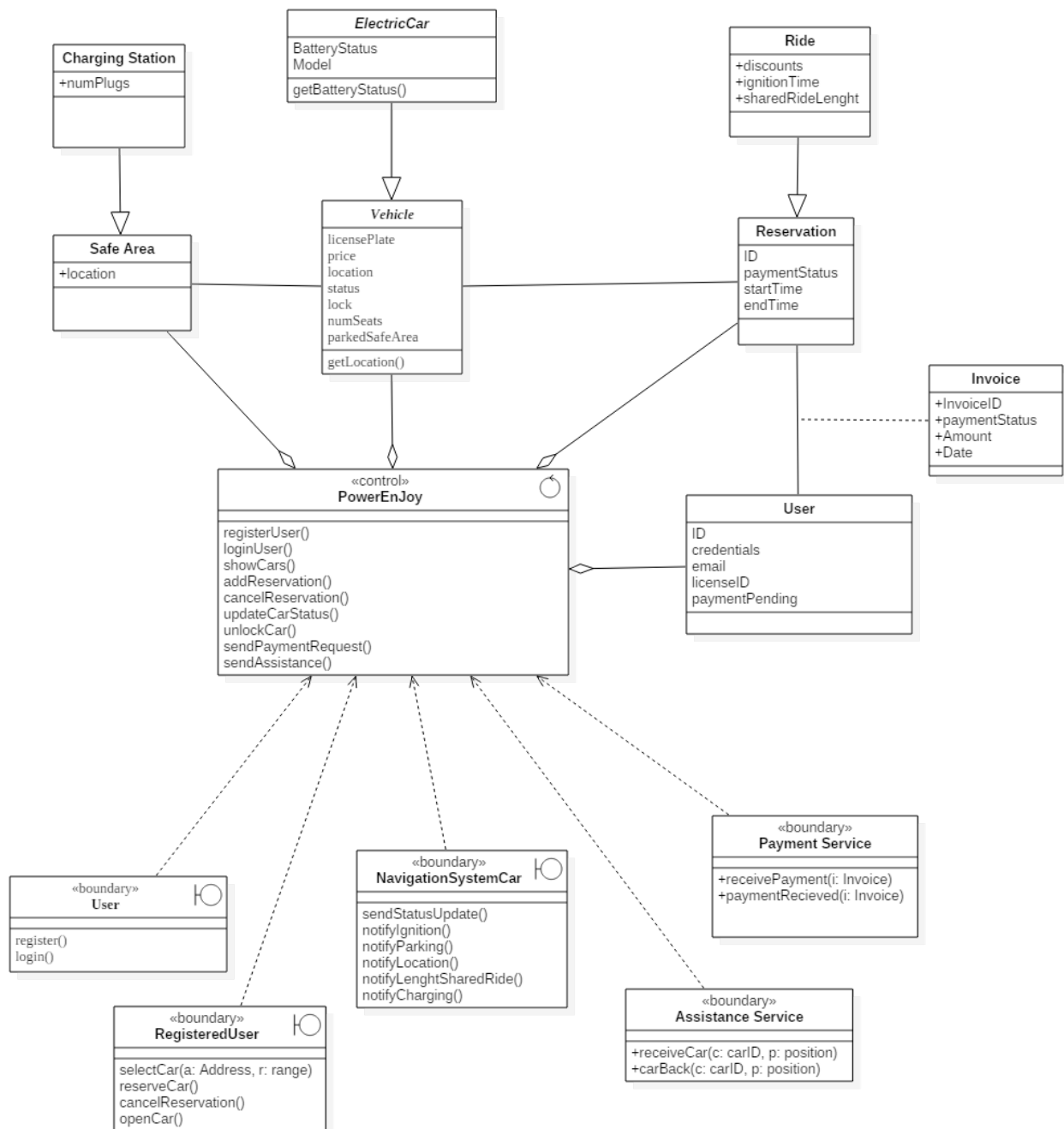
Entities: Objects representing system data, often from the domain model.

Boundaries: Objects that interface with system actors (e.g. a user or external service).

Controls: Objects that mediate between boundaries and entities. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions.

We choose this pattern for several reason:

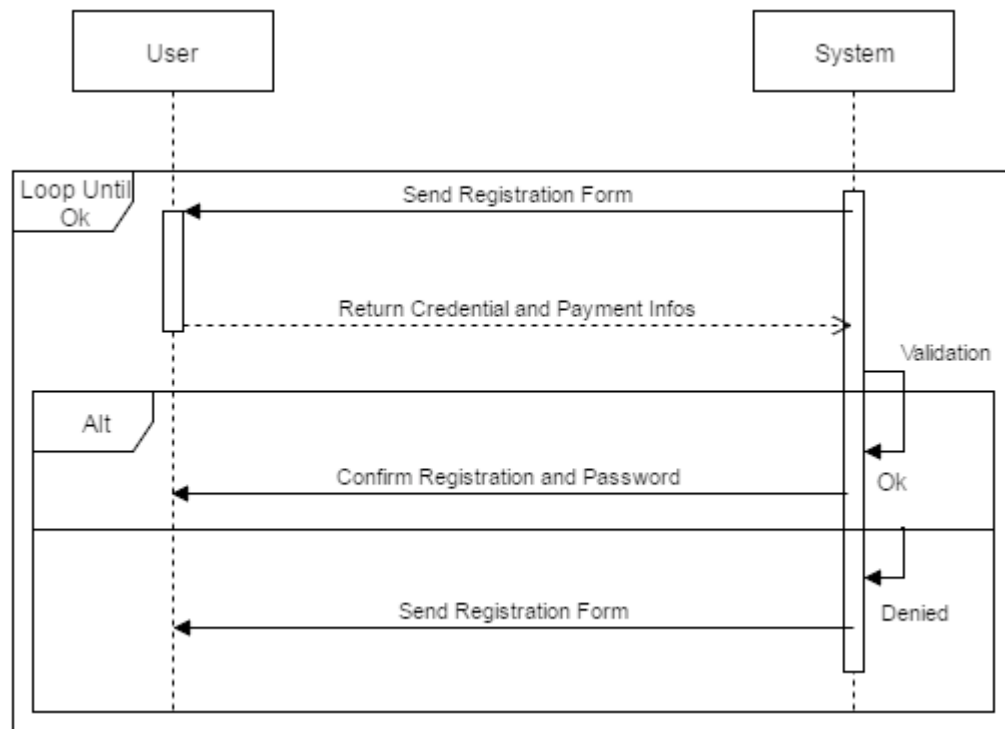
- To clearly separate what happens in the boundary from what happens "inside" our system
- Comparing the Use Case Diagram and the Class Diagram should be clear how the two are related
- It facilitate the analysis of cause / effects relations between phenomena in the world and in the machine (and viceversa)
- It will serve as reference for the next section where we analyze in details the single use cases



3 Functional Requirements

3.1 Use Case 1 [UC1]: Registration

- ❖ Participating Actors:
 - User
- ❖ Entry Condition:
 - A user notifies he wants to register to the system.
- ❖ Flow of Events:
 - User selects Registration.
 - The system asks him to insert payment informations and credentials.
 - The User inserts required data.
 - The system checks validity of the data.
 - The system generates an account for the User, associates given informations to the account, generates a password and gives it back to the User.
- ❖ Exit Condition:
 - This use case terminates when registration is successfully completed and the generated password is given back to the User.
- ❖ Exceptions:
 - If the payment information or the credentials are not validated the registration can't proceed, the User is notified the informations he provided aren't valid and is asked to provide them again.
 - Modifying credential and payment information is a particular case of registration, available only after the Login (UC2) by power users.
- ❖ Special Requirements:
 - Once the User has given his informations the System must reply within 5 minutes.



❖ Requirements

- System has a registration mechanism that allow users to insert their credentials.
- The system checks for duplicate users.
- The system checks for valid credential.
- The system has a modifying option that allow user to modify their credentials.

3.2 Use Case 2 [UC2]: Login

❖ Participating Actors

- User

❖ Entry Condition

- User selects the Login functionality

❖ Flow of Events

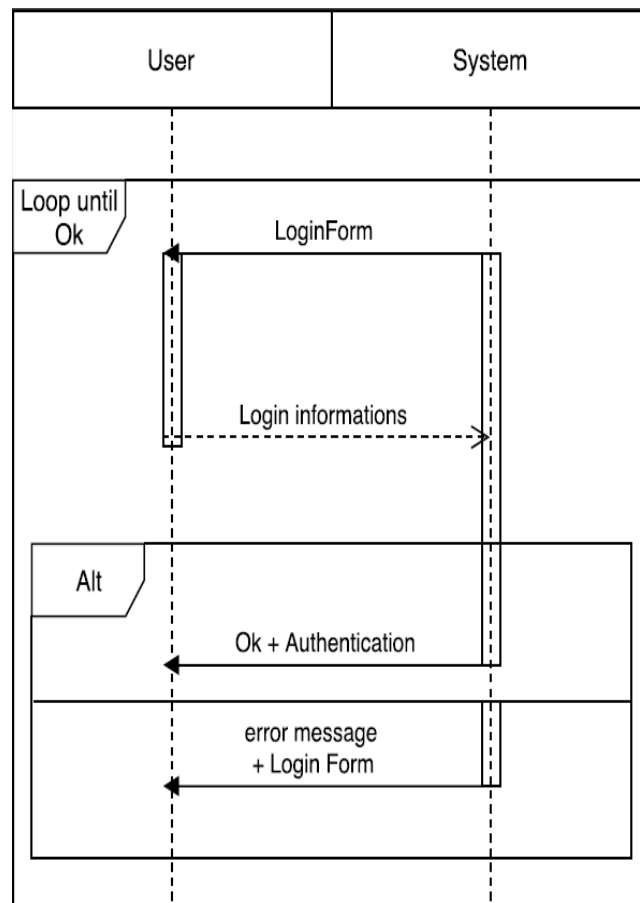
- The System requests the User to input his login informations.
- User inputs his login informations.
- The System executes a check and if the given informations are valid authenticate the User.

❖ Exit Conditions

- User is authenticated (User become PowerUser).

❖ Exceptions

- If the data provided by the user are not valid he is asked to input them again.



❖ Requirements

- System has a login functionality that allows users to enter their login informations (username and password).
- If provided information is correct, the system allows the user to access the service.

3.3 Use Case 3 [UC3]: Select Car

❖ Participating Actors:

- PowerUser

❖ Entry Condition

- A service requires the user to select a car.

❖ Flow of events

- The system checks the power user doesn't have pending payments.

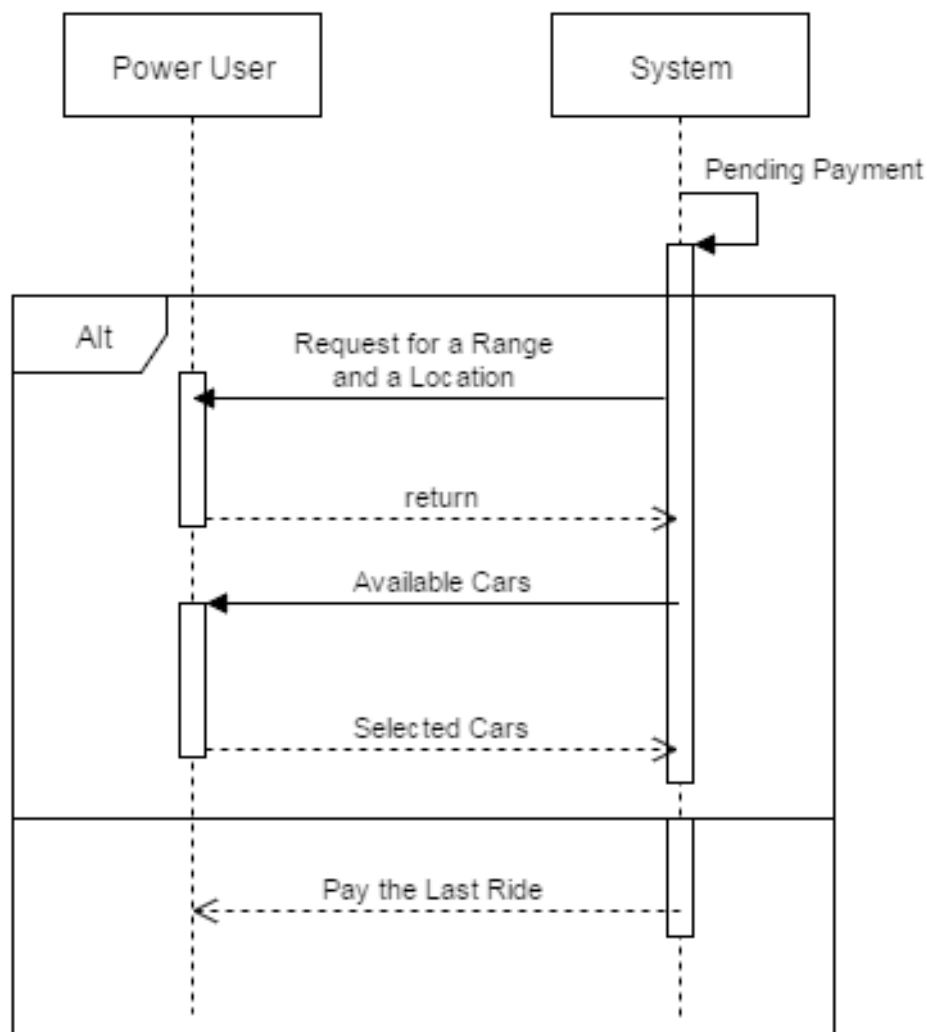
- The system asks the power user to select an address or to provide his gps location and to select a distance range.
- The power user inputs the required data.
- The system computes the list of all the available cars within the range from the given location and shows it to the user.
- The power user is shown for each car the distance from the user given location, the level of battery and whether or not it is charging.
- The power user selects a specific car and the identifier of the car is given back to the system.

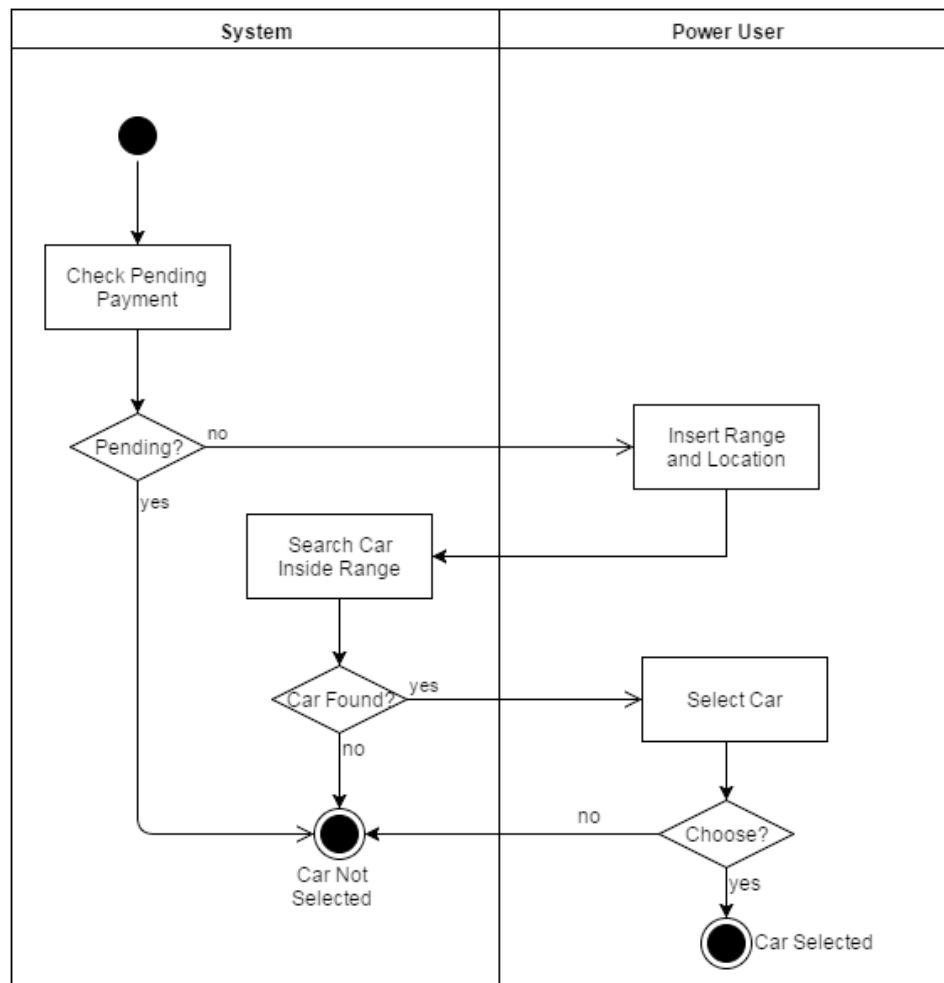
❖ Exit Condition

- The identifier of the selected car is given back to the system.
- The power user selects no cars.

❖ Exceptions

- If no such a car is available no car will be shown to the user.
- If there are pending payment on the power user, the system will ask to pay the debt (UC6).





❖ Requirements

- Power User must be able to specify a position and a distance range.
- System must be able to compute the distance between two positions.
- System must show a list of the suggested CARs that qualify as "AVAILABLE" and are in the specified range.
- System must allow the user to select a car from the “suggested car” list.

3.4 Use Case 4 [UC4]: Reserve Car

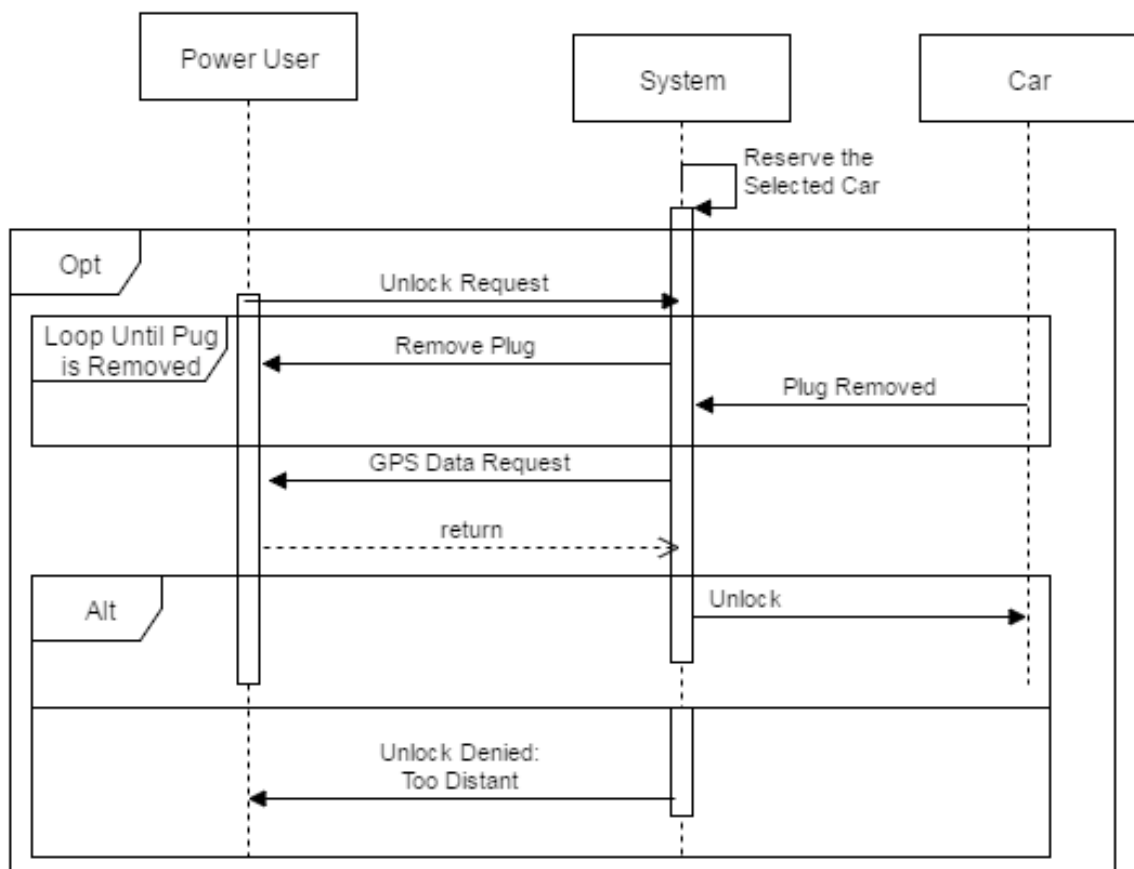
❖ Participating Actors:

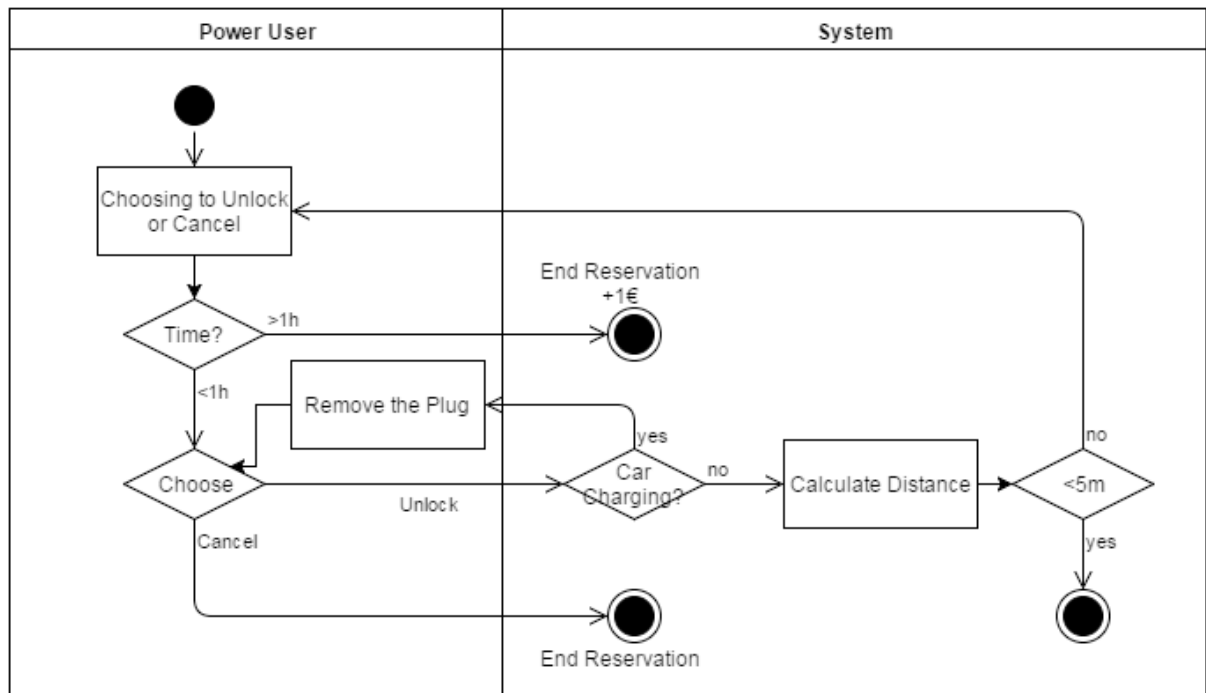
- Power User
- Car

❖ Entry Condition:

- A Power User selects a Car.

- ❖ Flow of events
 - The selected car is marked as reserved and a 1 hour timer is set.
 - The system waits for the user to select the unlock option.
 - When the power user selects the unlock option the system asks for his gps data.
 - The system computes the distance between user gps data and the reserved car gps data.
 - If this distance is below 5 meters the system sends an unlock command to the car.
- ❖ Exit Condition
 - The user starts the engine of the car.
- ❖ Exceptions
 - If after 1hr the power user hasn't already unlock the car, the reservation is canceled and a 1€ payment is accredited to the power user (UC6).
 - If the user selects the cancel reservation option, the reservation is canceled without conditions.
 - If the car is charging it will not open. The user have to remove the plug to open the car.





❖ Requirements

- The system should offer an option to the user to cancel the pending reservation.
- Reservations expire after one hour.
- The system should unlock the car only if the power user is within a certain distance.
- POWER User must be able to signal he wants the car he has reserved to be unlocked.
- When a reservation expires the POWER user who had it is charged 1 EUR
- The system should track the reservation time

❖ Special Requirements

- POWER USER can't have more than one reservation pending.

3.5 Use Case 5 [UC5]: Drive

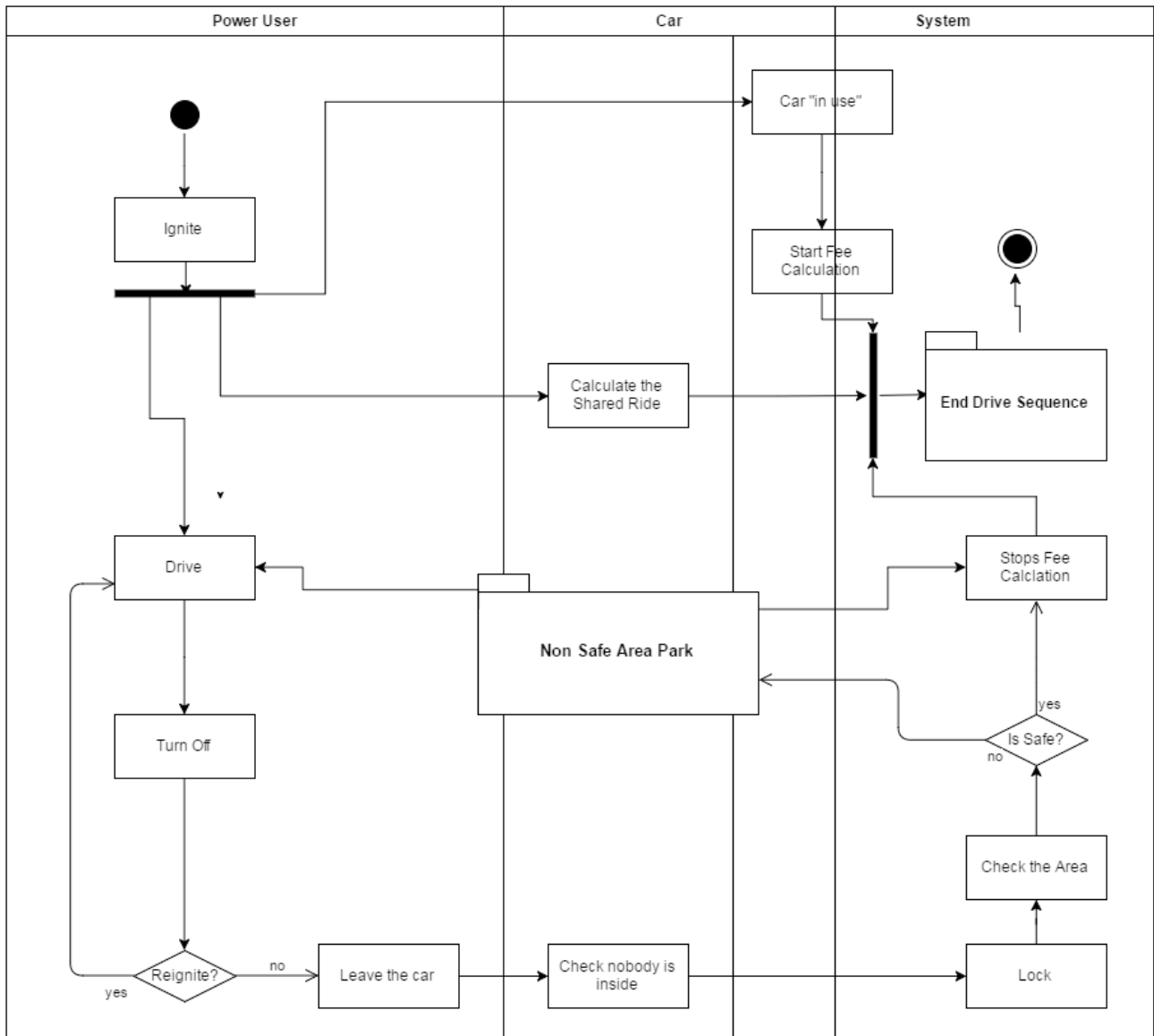
- ❖ Participating Actors
 - Power User
 - Car

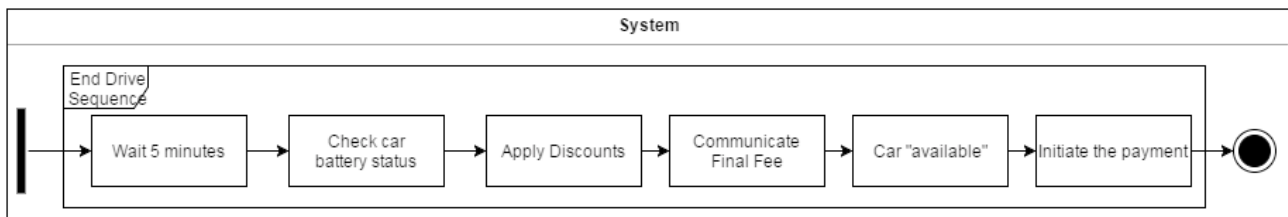
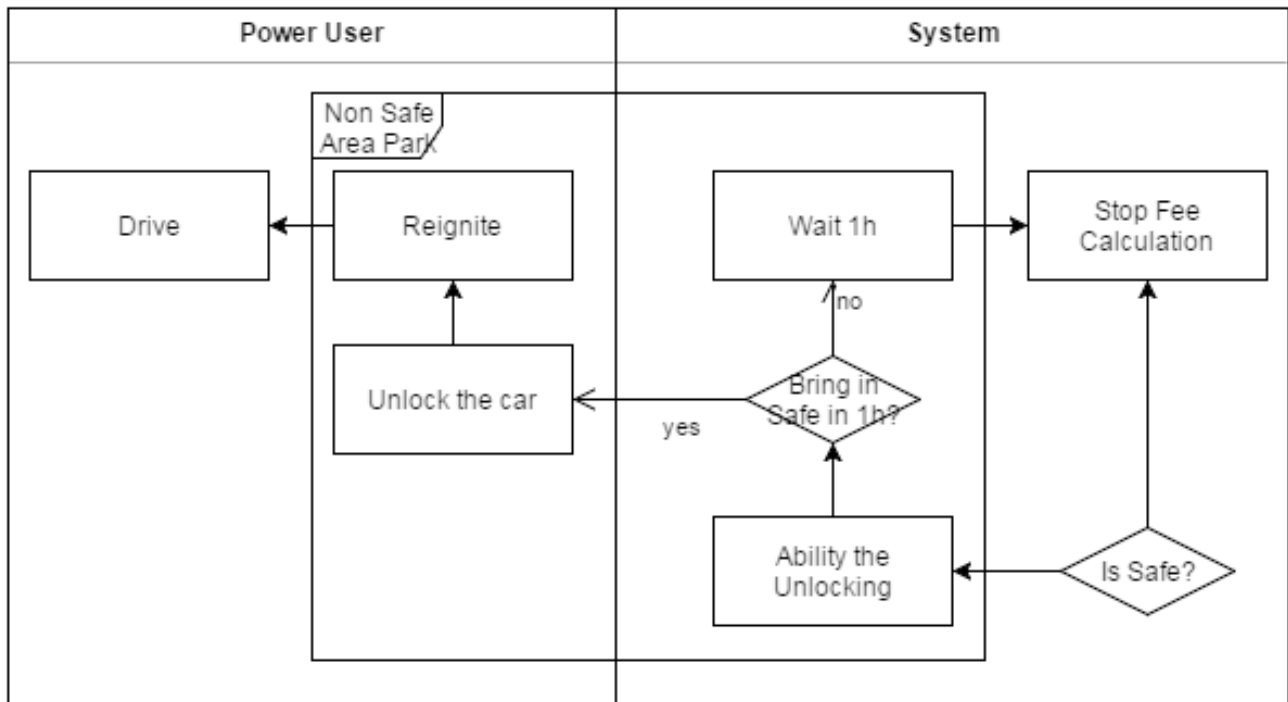
- ❖ Entry Condition
 - The car is ignited by the Power user

- ❖ Flow of Events:
 - The system put the car in “in use” state and starts the calculation of the fee (1€ per min).
 - The car starts/restart the shared ride calculation every time and until the passengers are two or more.
 - The power user starts driving.
 - The power user turn off the car.
 - The power user and all passengers leave the car.
 - The car check (by sensors) that nobody is in the car.
 - The system lock the car when nobody is inside.
 - The system check that the car is parked in a safe area.
 - The system stops the calculation of the fee.
 - The car send the shared ride value to the system.
 - The system waits 5 minutes.
 - The system applies discounts according to its policies.
 - The system communicate the total fee to pay to the power user.
 - The state of the car changes from “in use” to “available”.
 - The system starts the payment procedure (UC6).

- ❖ Exit Condition
 - The system starts the payment procedure (UC6).

- ❖ Exceptions
 - The car is parked and switched off in an unsafe area, the system maintain an “in use”state linked to the power user, the car will be locked when nobody is inside. The power user will have again the possibility to open the car (restart the use case). One hour after the engine stop, the system will close normally the rent.





❖ Requirements

- The RIDE FEE is computed in real time on a per minute base.
- The system must lock a car with no passenger inside and engine off.
- FINAL FEE will be calculated applying discounts/fines to the RIDE FEE according to the power enjoy discount policy.
- System notify the user of the FINAL FEE.

❖ Special Requirement

- The RIDE FEE for the ride must be constantly displayed in the car.
- The RIDE FEE is computed in real time on a per minute base

3.6 Use Case 6 [UC6]: Payment

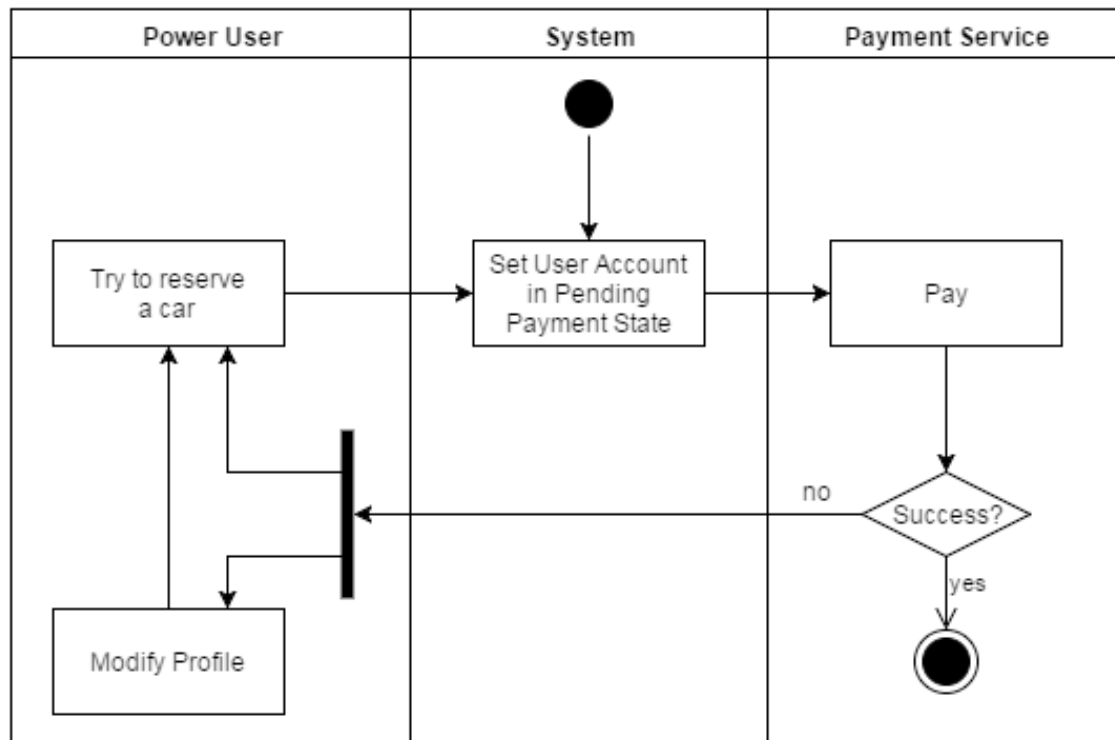
- ❖ Participating Actors
 - Payment Service

- ❖ Entry Condition
 - System has intention to charge a client.

- ❖ Flow of Events
 - The account of the user is set having a pending payment.
 - The System sends to the payment service a charge for the given amount and the user's payment informations.
 - The payment service signals that the payment has been accomplished.
 - The system sets the user as not having pending payments.

- ❖ Exit Condition
 - The user is set to not having pending payments.

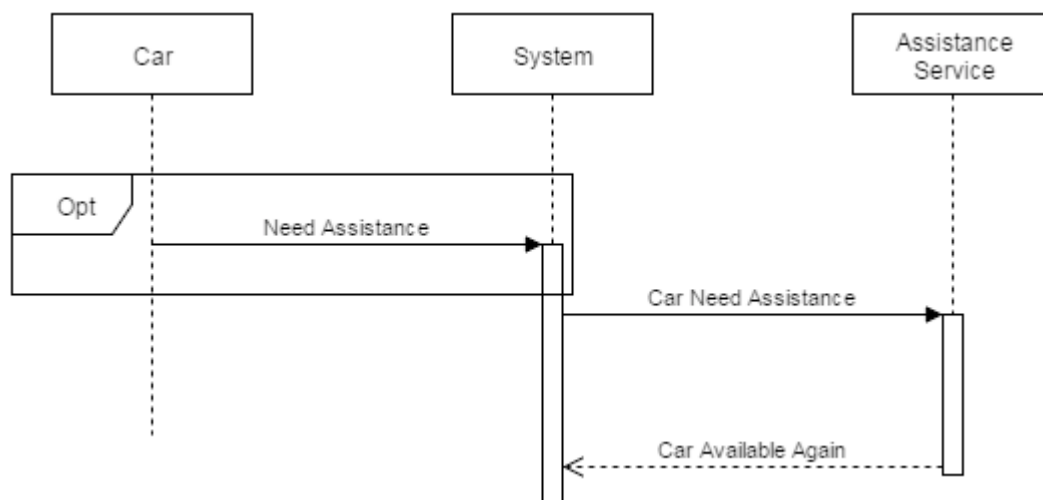
- ❖ Exceptions
 - If, for any reason, the payment can't be completed, the user will stay in a condition with a pending payment. A user with a pending payment can only modify his profile (Exceptions UC1) and try again to pay the debt.



- ❖ Requirements
 - POWER USERS should be prevented from reserving cars if they have pending payments
 - System should ask the payment service to process the power user payment
 - System should add a pending payment
 - System should notify the user of the pending payment.

3.7 Use Case 7 [UC7]: Require Assistance

- ❖ Participating Actors
 - Car
 - Assistance Service
- ❖ Entry Conditions
 - The car has any kind of malfunctioning
 - The car need to be brought back into a safe area.
 - The car need to be put on charge (less than 20% of the battery).
- ❖ Flow of Events
 - The car signals the need of assistance to the system.
 - The system makes the car unavailable.
 - The system signals the need of assistance, requested by the car, to the assistance service.
 - The assistance service solve the problem, according to the entry condition.
 - The assistance service notifies the system when the job is done.
 - The system updates the car informations (location and battery).
 - The system tags the car as available.
- ❖ Exit Condition
 - The car is available



3.8 Traceability Matrix

USER

G1. USER can register to the system with valid credentials

DX Every user is registered with a valid payment method to the external payment service.

DX The system is able to verify user credentials and external payment credentials

1. System has a registration mechanism that allow user to insert their credentials
2. The system checks for duplicate users
3. The system sends a password to access the system

G2. Only USER which are registered can login to the system

1. System has a login functionality that allows users to enter their login informations
2. If provided information is correct, the system allows the user to access the service.

POWER USER

G3. POWER USER can modify its personal informations

DX The system is able to verify user credentials and external payment credentials

1. System should allow POWER User to modify only their personal credentials
2. The system accepts the new credentials after validation

G4. POWER USER can see locations and battery levels of available CARs within a specific distance from a location (eg. POWER USER location or specified address)

D1. GPS always gives the correct position

D2. Every car has an always available GPS.

D4. Cars are able to detect current battery level.

D8. System knows the car status at any time.

D11. Every power user can be geolocalized.

1. POWER User must be able to specify a position and a distance range
2. System must be able to compute the distance between two positions
3. System must show a list of the CARs that qualify as "AVAILABLE" and are in the specified range

G5. POWER USER can reserve one available CAR.

D8. System knows the car status at any time.

1. System must allow the user to select a car from the list above
2. POWER USER can't have more than one reservation pending

G6. POWER USER can cancel a CAR RESERVATION within an hour before unlocking the car

D8. System knows the car status at any time.

1. The system should offer an option to the user to cancel the pending reservation
2. Reservations expire after one hour

G7. POWER USER can unlock his RENTED CAR

D1. GPS always gives the correct position

D2. Every car has an always available GPS.

D7 Cars with locked doors cannot be opened from the outside

D8. System knows the car status at any time.

D10. The system has remote control of the cars.

D11. Every power user can be geolocalized.

1. POWER User must be able to signal he wants the currently rented car to be unlocked
2. System should unlock the car only if the POWER USER is within a certain distance

G9. POWER USER knows the current RIDE FEE

D6 Every car is equipped with an onboard display.

D9. The system knows when the engine of a car has been ignited.

D8. System knows the car status at any time.

1. The RIDE FEE is computed in real time on a per minute base
2. The RIDE FEE for the ride must be constantly displayed in the car

G10. POWER USER knows the FINAL FEE

D2. Every car has an always available GPS.

D3. Cars are able to determine the number of passengers currently on board.

D4. Cars are able to detect current battery level.

D13. The set of safe areas and power grid stations is predefined

D15. The system knows when a car has been plugged for recharge

1. System notify the user of the FINAL FEE
2. FINAL FEE will be calculated applying discounts/fines to the RIDE FEE according to the power enjoy discount policy

G11. CARs are automatically locked when parked and the POWER USERS gets out

D3. Cars are able to determine the number of passengers currently on board.

D10. The system has remote control of the cars.

D9. The system knows when the engine of a car has been ignited/turned off.

D8. System knows the car status at any time.

1. The system must lock a car with no passenger inside and engine off.

Payment

G12. POWER USER with pending payments can't reserve cars

D18. Every user is registered with a valid payment method to the external payment service.

D19. Payment service takes care of the payment process and notifies when the payments occur

1. POWER USERS should be prevented from reserving cars if they have pending payments

G13. POWER USER is charged of the FINAL FEE

D19. Payment service takes care of the payment process and notifies when the payments occur

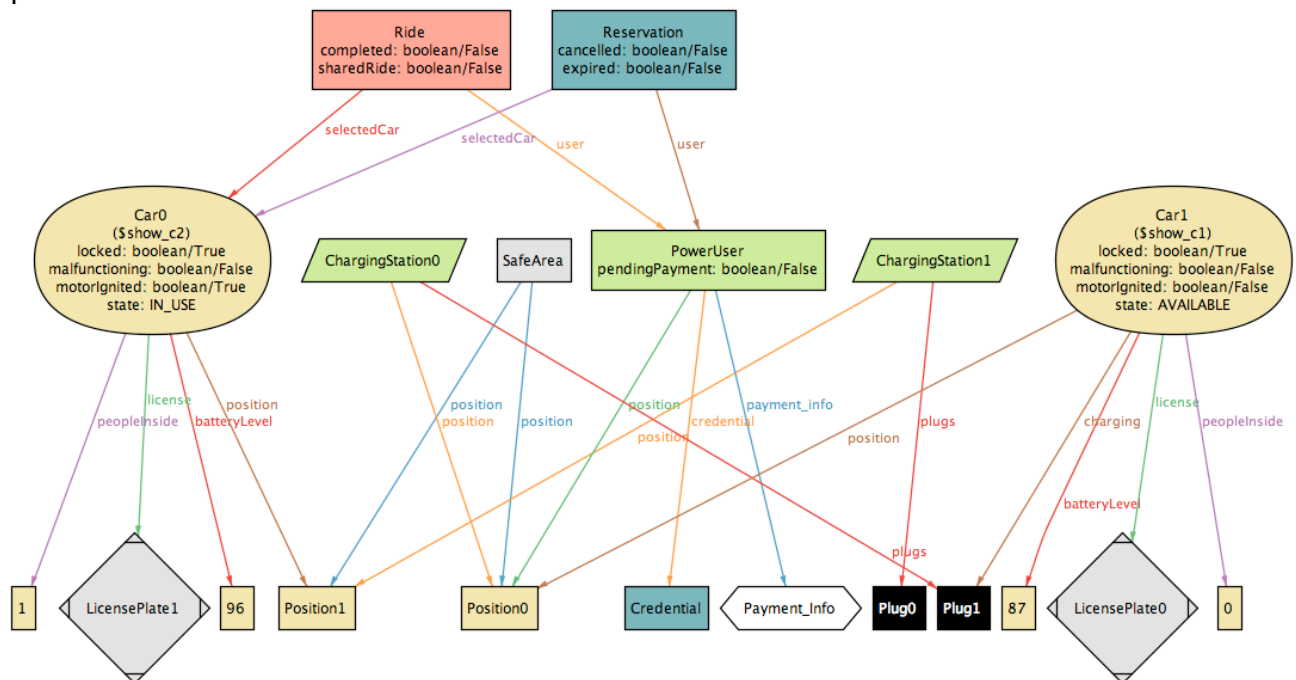
1. System should ask the PS to process the POWER USER payment
2. System should add a pending payment
3. System should notify the user of the pending payment

G14. If a car is not picked-up within one hour from the reservation the user pays a FEE of 1 EUR.

1. System should keep track of the reservation time

4 Non Functional Requirements

- Privacy: the system should guarantee the privacy of all the user's informations, nobody out of the system can collect informations that are stored in the Power Enjoy company database.
- Security: the system is able to protect himself from hacking attacks.
- Portability: various version of the app were released. The purpose is to makes the Power Enjoy service available from any kind of portable device with the internet access and a GPS system.
- Usability: the understanding of the app, and each section of it, is easy and intuitive.
- Accessibility: the app give to the user the possibilità to select language. The main language is english; italian, french, spanish, german, russian, chinese and japanese are also available.
- The system will be available constantly to guarantee the power enjoy constantly. Manutention sections are exceptions and should be preventively notified.
- Scalability: the system should guarantee notable performances with a growing number of active users.
- The system should receive, process and send messages to guarantee certain performances.



5. Alloy

5.1 Code

```

open util/boolean

sig Position {}
sig Credential {}
sig Payment_Info {}
sig LicensePlate {}
sig Plug {}

one sig SafeArea {
    position: some Position
}

sig ChargingStation {
    position: one Position,
    plugs: some Plug,
}

sig PowerUser {
    credential: Credential,
    payment_info: Payment_Info,
    position: one Position,
    pendingPayment: one Bool,
}

abstract sig CarState {}
one sig AVAILABLE extends CarState {}
one sig RESERVED extends CarState {}
one sig IN_USE extends CarState {}
one sig UNAVAILABLE extends CarState {}

sig Car {
    batteryLevel: Int,
    peopleInside: Int,
    motorIgnited: Bool,
    malfunctioning: Bool,
    license: one LicensePlate,
    position: one Position,
    locked: one Bool,
    charging: lone Plug,
    state: one CarState
}
{
    batteryLevel >= 0
    batteryLevel <= 100
    peopleInside >= 0
    peopleInside <= 5
}

sig Reservation {
    user: one PowerUser,
    selectedCar: one Car,
    invoice: lone Invoice,
}

```

```

        expired: one Bool,
        cancelled: one Bool,
    }
    {
    expired = True implies (no r: Ride | r.user = user and r.selectedCar =
    selectedCar)
        cancelled = True implies (no r: Ride | r.user = user and r.selectedCar =
    selectedCar)
        expired = True <=> invoice != none
        invoice != none implies invoice.user = user
    }

sig Invoice{
    user: one PowerUser,
    discount: lone Int,
    fine: lone Int,
    paid: one Bool,
}

sig Ride {
    user: one PowerUser,
    selectedCar: one Car,
    invoice: lone Invoice,
    completed: Bool,
    sharedRide: Bool,
}
{
    sharedRide = True <=> selectedCar.peopleInside >= 3
    completed = False implies invoice.discount = none
    completed = False implies invoice.fine = none
    completed = True <=> invoice != none
    invoice != none implies invoice.user = user
}

// USER FACTS
fact userFacts {
    all u1, u2: PowerUser | u1 != u2 <=> u1.credential != u2.credential
    all u1, u2: PowerUser | u1 != u2 <=> u1.payment_info != u2.payment_info
    PowerUser.payment_info=Payment_Info
    PowerUser.credential = Credential
}

// USER WITH PAYMENT PENDING
fact PaymentPendingUser {
    all u: PowerUser | u.pendingPayment = True <=> one i: Invoice | i.user = u
    && i.paid = False
    all u: PowerUser | u.pendingPayment = True implies (no r: Reservation |
    r.user = u && r.expired = False && r.cancelled = False)
}

// INVOICES
fact ValidInvoices {
    no disjoint r1, r2: Ride | r1.invoice = r2.invoice
    no disjoint r1, r2: Reservation | r1.invoice = r2.invoice
    all i: Invoice | i != none implies (one r: Reservation | r.invoice = i) or
    (one rr: Ride | rr.invoice = i)
}

// CAR

```

```

fact carLicenseUnique {
    all c1, c2: Car | (c1 != c2) => c1.license != c2.license
    #LicensePlate = #Car.license
}

fact carPositionUnique {
    all c1, c2: Car | (c1 != c2) => c1.position != c2.position
}

fact carLockedWhenNoOneInside {
    all c: Car | c.peopleInside = 0 implies c.locked = True
}

fact motorIgnitedOnlyWithPeopleInside {
    all c: Car | c.motorIgnited = True implies c.peopleInside > 0
}

fact motorIgnitedImpliesRide {
    all c: Car | c.motorIgnited = True implies (one r: Ride | r.selectedCar =
c)
    all c: Car | c.motorIgnited = True implies c.state = IN_USE
}

// CAR STATES

fact carAvailableState {
    all c: Car | c.state = AVAILABLE implies c.batteryLevel >= 20
    all c: Car | c.state = AVAILABLE implies c.motorIgnited = False
    all c: Car | c.state = AVAILABLE implies c.peopleInside = 0
    all c: Car | c.state = AVAILABLE implies (one s: SafeArea | c.position in
s.position) or (one ch: ChargingStation | c.position = ch.position)
}

fact carReservedState {
    all c: Car | c.state = RESERVED => one r: Reservation | r.selectedCar = c
    all c: Car | c.state = RESERVED implies no r: Ride | r.selectedCar = c
    all c: Car | c.state = RESERVED implies c.malfunctioning = False
    all c: Car | c.state = RESERVED implies c.motorIgnited = False
    all c: Car | c.state = RESERVED implies c.batteryLevel >= 20
    all c: Car | c.state = RESERVED implies (one s: SafeArea | c.position in
s.position) or (one ch: ChargingStation | c.position = ch.position)
}

fact carInUseState {
    all c: Car | c.state = IN_USE <=> one r: Ride | r.selectedCar = c
    all c: Car | c.state = IN_USE implies one r: Reservation | r.selectedCar =
c && r.expired = False && r.cancelled = False
    all c: Car | c.state = IN_USE implies c.peopleInside > 0
}

fact carUnavailable {
    all c: Car | c.state = UNAVAILABLE implies no r: Reservation |
r.selectedCar = c
    all c: Car | c.state = UNAVAILABLE implies c.locked = True
    all c: Car | c.state = UNAVAILABLE <=> c.malfunctioning = True
    all c: Car | c.state = UNAVAILABLE <=> c.batteryLevel <= 20 &&
c.motorIgnited = False && c.peopleInside = 0
    all c: Car | c.state = UNAVAILABLE <=> c.batteryLevel = 0
}

```

```
// RESERVATION

fact oneReservationPerUser{
    no disjoint r1, r2: Reservation | r1.user = r2.user
}
fact oneReservationPerCar {
    no disjoint r1, r2: Reservation | r1.selectedCar = r2.selectedCar
}

fact ReservationNotCompletedAndExpired {
    no r: Reservation | r.expired = True && r.cancelled = True
}
// RIDE
fact NoRideWithoutActiveReservation {
    all r: Ride, c: Car | r.selectedCar = c implies one rr: Reservation |
rr.user = r.user and rr.selectedCar = r.selectedCar and rr.expired = False &&
rr.cancelled = False
}

fact onlyOneReservationPerCar {
    no disjoint r1, r2: Reservation | r1.selectedCar = r2.selectedCar
}

// ChargingStation
fact ChargingStationAreSafeArea {
    all ch: ChargingStation | ch.position in SafeArea.position
}

fact carsCannotBePluggedInTheSamePlug {
    no disjoint c1, c2: Car | c1.charging = c2.charging
}

fact ChargingStationUniquePosition {
    no disjoint c1, c2: ChargingStation | c1.position = c2.position
}

fact noChargingVehicle {
    all c: Car | c.motorIgnited = True implies c.charging = none
    all c: Car | c.locked = False implies c.charging = none
    all c: Car | c.peopleInside > 0 implies c.charging = none
}

fact plugBelongsToAChargingStation {
    all p: Plug | p != none implies (one c: ChargingStation | p in c.plugs)
    no disjoint c1, c2: ChargingStation | c1.plugs & c2.plugs != none
}

fact carsChargingSamePositionOfChargingStation{
    all c: Car | c.charging != none implies (one cs: ChargingStation |
c.position = cs.position)
}

fact carsChargingAssignedToAPlug{
    all c: Car | c.charging != none implies (one cs: ChargingStation |
c.charging in cs.plugs)
}

// discount
```

```

fact reservationInvoiceHaveNoDiscount {
    all r: Reservation | r.invoice.fine = none and r.invoice.discount = none
}

fact NoDiscountIfFine {
    all r: Ride | r.invoice.fine != none implies r.invoice.discount = none
}

fact discount {
    all r: Ride | r.completed = True && r.selectedCar.state = IN_USE &&
r.sharedRide = True implies r.invoice.discount = 10
    all r: Ride | r.completed = True && r.selectedCar.state = IN_USE &&
r.selectedCar.batteryLevel > 50 implies r.invoice.discount = 20
    all r: Ride | r.completed = True && r.selectedCar.state = IN_USE &&
r.selectedCar.charging != none implies r.invoice.discount = 30
}

fact fines {
    all r: Ride | r.completed = True && r.selectedCar.state = IN_USE &&
r.selectedCar.batteryLevel <= 20 implies r.invoice.fine = 20
}

/* ASSERT

assert noRideForUserWithDebts {
    all r: Ride | r.completed = False and r.user.pendingPayment = False
}
check noRideForUserWithDebts

assert allCompletedRidesHaveInvoice {
    all r: Ride | r.completed = True and r.invoice != none
}
check allCompletedRidesHaveInvoice

assert allRunningCarsHaveActiveRide {
    no c: Car | c.state = IN_USE and (no r: Ride | r.selectedCar = c and
r.selectedCar.state = IN_USE) }
check allRunningCarsHaveActiveRide

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

assert noCarReservedMoreThanOnce {
    no disjoint r1, r2: Reservation | r1.selectedCar = r2.selectedCar }
check noCarReservedMoreThanOnce

assert noUnreservedCarInReservation {
    all r: Reservation | r.selectedCar.state = RESERVED }
check noUnreservedCarInReservation

*/

pred show () {}

run show for 2 but 10 Int

```

5.2 Asserts

```
Executing "Check carleftinasafearea"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7406 vars. 432 primary vars. 13869 clauses. 466ms.
  No counterexample found. Assertion may be valid. 59ms.
```

```
Executing "Check CarAlwaysLeftInASafeArea"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7406 vars. 432 primary vars. 13869 clauses. 181ms.
  No counterexample found. Assertion may be valid. 26ms.
```

```
Executing "Check noRideForUserWithDebts"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7399 vars. 432 primary vars. 13873 clauses. 100ms.
  No counterexample found. Assertion may be valid. 13ms.
```

```
Executing "Check noUnreservedCarInReservation"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7398 vars. 432 primary vars. 13873 clauses. 42ms.
  No counterexample found. Assertion may be valid. 5ms.
```

```
Executing "Check noCarReservedMoreThanOnce"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7426 vars. 435 primary vars. 13936 clauses. 49ms.
  No counterexample found. Assertion may be valid. 4ms.
```

```
Executing "Check noUserWithMoreThanOneReservation"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7426 vars. 435 primary vars. 13936 clauses. 45ms.
  No counterexample found. Assertion may be valid. 4ms.
```

```
Executing "Check allRunningCarsHaveActiveRide"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7421 vars. 432 primary vars. 13891 clauses. 40ms.
  No counterexample found. Assertion may be valid. 0ms.
```


5.3 Worlds Generated

