POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

RASD: Requirements Analysis and Specification Document

Author(s): **Lorenzo Ferretti**
Lorenzo Manoni
Carlo Sgaravatti

Version: 1.1

Date: 08/01/2023

Academic Year: 2022-2023

Contents

1 Introduction

1.1. Purpose

The scope of this document consists of showing the requirements of the system to be. Since the system to be is made up of two different sub-systems the eMSP and the CPMS, that also have different final users, this document will include a description of both systems. Characteristics of the system are described following a top-down approach with the aim to facilitate comprehension of the readers.

1.1.1. Goals

User goals:

G1 Allow users to visualize nearby charging stations and the relative price of energy, socket availability, and if they provide offers;

G2 Allow users to reserve a charge at a certain charging point for a certain time slot;

G3 Allow the user to start a charging process at a station;

G4 Allow the user to monitor the status of the charging process with valuable data;

G5 Allow the user to pay for the service;

G6 Gives users suggestions about the optimal schedule to charge the vehicle depending on battery status, the schedule of the users, and offers provided by CPOs;

CPO goals:

G7 Allow CPOs dynamically decide from which DSO to acquire energy providing information on energy prices;

G8 Allow CPOs dynamically decide the cost of charging and when setting special offers;

G9 Allow CPOs dynamically decide whether or not to store energy in their internal batteries;

G10 Allow CPOs dynamically decide, during a charging process, to use the stored energy in the battery or to acquire directly from DSOs to fulfil the charge or a mix of both;

G11 Allow the CPOs to Manage their CPs and the relative socket in order to enable drivers to use them.

1.2. Scope

Recent technological advances make electric vehicles (EVs) a plausible alternative to gasoline vehicles and give them the potential to enable zero or low-emission transportation. Improved batteries are cheaper and permit longer driving ranges making EVs more attractive to consumers than in the past. Achieving zero emissions is a goal shared by multiple countries and new green technologies are encouraged all around the world. Investments in Green transition energy projects are estimated to value 257 billion dollars.

Considering that the Charging Point infrastructure is a crucial asset to succeed in the EV strategy, our product aims to optimally manage the complex infrastructure needed for electric mobility. The scope of the project aims to satisfy both driver and Charging Point Operators needs. Drivers need new means to feel comfortable with Electric Mobility, "range out anxiety" has been defined as one of the main issues that obstacle to the adoption of Electric Vehicles in many parts of the world. One of our major goals is to improve EV' drivers' experience through a mobile application, the eMSP, that can help drivers to feel more comfortable, empowering users by informing them about all the charging points they can access and allowing them to reserve charging sessions at a Charging Point. To improve drivers' experience the system actively makes suggestions of stops during a trip using the navigation system and is able to access users' calendars to prone charging sessions that fit the user schedule at best. The system assists the driver also during the charging sessions, providing information on the charging process and notifying it when the process is complete. The user can also benefit from our payment system to pay in a rapid and smart way to pay.

This is not the only aspect of the EV infrastructure that we care about, in fact, the system also provides software for the management of the CP. This software is the CPMS and the scope of this product is to help the Charging Point Operators (CPOs) to manage their Charging Points. We provided all crucial functionality for all operations executed by them. We support the acquisition of energy by different energy providers (DSOs), we provide a system to manage each Charging Point allowing the CPOs to monitor the status of each socket, and the status of the battery if present and to perform all critical operations on them (like the exclusion of the battery from the system, the management of

the stock availability and the socket charging profile). Our system doesn't limit to offering industry-standard operations, in fact, it has an intelligent component for optimizing energy acquisition and management. This is a crucial feature that enables CPOs to reduce costs through a cost-optimal energy choice, which can result to be a really important asset in a competitive scenario. Another key feature of the system is to give to CPOs the possibility to create special offers to the customer that can directly access them through the eMSP.

Both the eMSP and the CPMS can interact with all the other systems provided by others.

1.2.1. World Phenomenas

WP1 The user arrives at the CP with his electric car;

WP2 The user sets a route in the navigation system;

WP3 The user uses a calendar to organize a personal schedule;

WP4 The user plugs the socket;

WP5 The user unplugs the socket.

WP6 The user drives his electric car.

WP7 The CP locks a socket.

WP8 The CP unlocks a socket.

WP9 The CP starts the energy flow to the connected vehicle.

WP10 The CP stops the energy flow to the connected vehicle.

1.2.2. Shared Phenomenas

	Phenomena	Controller
SP1	The user registers to the eMSP	W
SP2	The user logins to the eMSP	W
SP3	The user checks the CP nearby	W
SP4	The user checks for the offers	W
SP5	The user reserves a CP	W
SP6	The user checks the status of the charging process	W
SP7	The user stop the charging process	W
SP8	The user ends the charging process	W
SP9	The user pays for the service	W
SP10	The user checks for the offers	W
SP11	The CPO selects the source of energy	W
SP12	The CPO manages the offers	W
SP13	The CPO register a new CP in the CPMS	W
SP14	The CPMS detects a plugged socket through the CP	W
SP15	The CPMS detects a unplugged socket through the CP	W
SP16	The eMSP notifies the user when the charging process is completed	M
SP17	The eMSP retrieves information from the user's calendar	M
SP18	The eMSP retrieves information from the user navigation system	M
SP19	The eMSP retrieves information from the car battery	M
SP20	The eMSP notifies the user of suggestions for the recharging process at a certain CP and time frame	M
SP21	The CPMS communicates to the CP to lock/unlock a socket	M
SP22	The CPMS communicates to the CP to start/end the charging process	M
SP23	The CPMS acquires information about the CPs (socket status, battery status, position, charging processes)	M
SP24	The CPMS acquires information about energy prices and availability from DSOs	M
SP25	The CPMS changes the source of energy to recharge vehicles	M
SP26	The CPMS communicates to the CP to store the energy in the battery	M

Table 1.1: Shared Phenomena

1.3. Glossary

1.3.1. Definitions

- **Charging Point:** it's the physical system where an electric vehicle can be recharged.
- **Charging Point Operator:** it's a company that owns some charging points.
- **Distribution System Operator:** it's a company that provides energy to the charging points.
- **Socket:** it's a physical connector of the charging point on which a vehicle is connected when recharging. There are three types of physical connection for the sockets: slow, fast and rapid. Each of those types specifies a maximum value of energy flow that the socket is able to provide to the connected vehicles.
- **Charging Profile:** it's a schedule that defines how much energy a particular socket will provide to the connected vehicle during the charging process. Usually, the schedule is composed of a list of different periods (e.g. different time intervals of a day) on which the amount of energy that the socket will provide can be different.
- **User:** it's a person who is registered to the service and that can make reservations in some charging points in order to recharge his vehicle.
- **Charging Session:** it's the process by which a vehicle is being recharged by a certain socket.
- **Notification:** it's an alert that the eMSP sends to the user to inform him of a certain event.
- **Tariff:** it's the price, specified in terms of units of energy, for recharging the vehicles in a certain charging point. Different types of sockets of the same charging point can have different tariffs.
- **Special Offer:** it's a particular tariff that is valid only for a certain time slot.
- **Current Energy Source:** it's a source of energy that is providing energy to a charging point for recharging vehicles. A charging point can have more than one current energy source, i.e. a DSO and one or more batteries.
- **Energy Mix:** it's the way in which the charging point is using its current energy sources. Given the amount of energy demanded, the Energy mix specifies the relative percentage of energy that the charging point should try to satisfy using the battery/batteries.

- **Manual/Automatic mode:** the way in which the CPMS is used to manage a charging point. There are three functions that can have one of the two modalities: the DSO selection, the energy mix management and the tariff policy management. If the CPMS is in automatic mode for a certain charging point and for a certain function, it automatically optimizes that function without the need for manual intervention. In manual mode, the CPMS allow the CPO to manually manage the configurations.
- **Battery level:** it's the actual amount of energy stored inside a battery of a charging point.
- **Battery Minimum Energy Trigger:** it's a value that indicates the minimum battery level at which the battery can be used as an energy source. When the CPMS is set on manual mode for energy management and a battery reaches that value the CPMS automatically uses another energy source to recharge it.
- **Battery Maximum Energy Trigger:** it's a value that indicates the maximum battery level at which the battery can be used as an energy source. When the CPMS is set on manual mode for energy management and a battery reaches that value the CPMS automatically uses the battery to recharge the plugged cars. This value must be higher or equal to the Minimum Energy trigger.
- **Battery availability:** a battery can be available or unavailable. Only in the first case, it can be used as an energy source for the charging point.
- **Battery status:** identifies how the battery is working at a given time when it is available. The battery can have the following status:
  - Export: the battery is used as an energy source for the charging point.
  - Import: the battery is being recharged by another energy source, i.e. a DSO.
  - Idle: the battery isn't used as an energy source and isn't being recharged.

1.3.2. Acronyms

- **CPMS:** Charging Point Management System
- **eMSP:** e-Mobility Service Provider
- **OCPI:** Open Charge Point Interface, which is the communication protocol used to connect eMSP and CPMS.

- **OCPP:** Open Charge Point Protocol, which is the communication protocol used to connect the CPMS and the charging point.
- **API:** Application Programming Interface, it's a way by which two software systems communicate.

1.3.3. Abbreviations

- **CP:** Charging Point
- **CPO:** Charging Point Operator
- **DSO:** Distribution System Operator
- **Gn:** Goal number n
- **WPn:** World Phenomena number n
- **SPn:** Shared Phenomena number n
- **Dn:** Domain assumption number n
- **UCn:** Use Case number n
- **Rn:** Requirement number n

1.4. Revision history

- December 23, 2022: version 1.0 (first release)
- January 8, 2023: version 1.1
  1. Modified StartChargingSession and EndChargingSession use cases
  2. Eliminated 7 requirements regarding the unlock of a socket and, as a consequence, fixed the mapping

1.5. Reference Documents

- Project assignment ad specification document "Assignment RDD AY 2022-2023_v3.pdf"
- OSCP 2.0 Specification.pdf
- SMUD OpenADR Implementation Design Guide v1_0.pdf
- ocpp-1.6.pdf

- OCPI-2.2.1.pdf

1.6. Document Structure

The document's structure is designed with the aim to introduce gradually the reader into the system leading to an increased detail level, for this reason, it has been decided to follow a top-down approach in the description of the problem and requirements. After the first chapter of the introduction, an entire section is dedicated to the introduction of the problem.

The aim of that section consists in to give the reader a general perspective of the product in question(section 2.1) providing a description of the domain through a class diagram, and a state diagram and describing the main different scenarios. But also provides a high-level perspective of the system's functionality (section 2.2) through the product functions description, a description of the different users that will interact with the system (section 2.3), and the assumption taken to model and design the system (section 2.4).

The third section is dedicated to providing a more detailed description of the requirements. First (section 3.1) are described the External Interface Requirements that include the Hardware, Software, Communication, and User interface. Then an entire section (section 3.2) is dedicated to the functional requirements including a detailed description of all use cases, to better help the reader's understanding, each use case comes with a sequence diagram. Then (section 3.3) the performance requirements are described. Section 3.4 is about the design constraint of the system, which describes the Standard Compliance and the Hardware Limitations. The last part (section 3.5) is dedicated to the Software System Attributes, there is described the desired performance that the system should satisfy in terms of Reliability, Availability, Security, Maintainability, and Portability. Each one of these aspects has exposed the criticality of it for the success of the application.

The fourth section is entirely dedicated to the formal analysis of the requirements using alloy, in order to validate their completeness and consistency of them.

Then the last two sections describe the effort spent by the team to model the requirements and write the document and the last one included all the references consulted to write the document.

2 | Overall Description

2.1. Product perspective

2.1.1. Scenarios

1. **A new user wants to start using the eMall service.** The user Luke has just bought his new electric vehicle and wants to find places where he can charge the battery of his new car. Luke decides to download the eMall service on his mobile phone and register for it. He opens the applications and inserts his data, including his car information and a payment method; he also allows the system to access his calendar, his position, and navigation system in order to use all features provided.

2. **A user inserts his vehicle's information.** John is a new user really passionate about electric cars, in his garage has 4 different electric vehicles. During the registration, he inserts all his cars, provides the related information, and selects the car that uses on a daily basis as his "Favourite Vehicle". The eMSP gives John suggestions considering his "Favourite Vehicle", also when John books a charge, the system considers only the sockets compatible with the car. On the weekend John decides to drive another car and selects his truck as his "Favourite Vehicle".

3. **A user receives a suggestion based on the navigation system.**
   The user Luke is driving for a long trip and his navigation system is active. While he is following the path, the system acquires information on the battery status and realizes that he will not be able to reach the destination without stopping for charging and it computes the best charging station to stop. The system sends a notification to Luke about the suggested stop for the "Favourite Vehicle"; Luke accepts the suggestion and the system reserve a socket of the correct type for his car in the suggested charging point on the path of the navigation system.

4. **User recharges his vehicle and pays for the service.**
   Luke has already booked a charge in the charging station Y, arrives at the location, and parks the car in front of the correct socket. From the mobile application, he

selects the option to unlock the socket. The CPMS unlocks the socket, Luke inserts the plug in the socket and clicks on the start charging process option. The CPMS locks the socket and considers all available options to transfer the energy considering the cost of the electricity of every DSO available and the CP battery, at the end it decides to use the CP battery for the energy flow and starts the charging process. After some minutes the CP detects that the battery of the car is full and stops the energy flow sending an update to the CPMS. The CPMS informs the eMSP that sends a notification to Luke, who returns to the car. Luke ends the charging process with his mobile application, the CPMS unlocks the socket and Luke removes the plug. In the end, the eMSP pays for the service with the payment method provided by Luke. Now the socket is free and available for other users.

5. **CPO chooses a DSO as the energy source for a CP.**
   Mark works for a CPO, recently the company signed a contract with "Energy Distribution" DSO that force the company to acquire the energy only from it for an entire month. Mark login into the CPMS of the company with his credentials and select the option that forces every CP to acquire energy only from "Energy Distribution". At the end of the month, Mark reset the CMPS to let it choose the best strategy to acquire energy.

6. **CPO includes the battery as an energy source for a CP.**
   Mark, that still works for a CPO, through the CPMS must do some maintenance work for increasing the CP battery capacity, so decides to manually set the energy acquisition only from DSOs, excluding the CP battery. After a week the CP battery is repaired and Mark resets the default settings, in this way the CPMS has full control of the CP. The CPMS notices that the prices of the DSO are increased exponentially in this period and the CP battery is not used to its new full potential, so mix the energy flow with the DSOs, in order to optimize the cost.

7. **User receives a suggestion based on the calendar.**
   Luke uses his calendar on a daily basis to manage his business meetings. The eMSP acquires information from his mobile phone calendar application, considers also the battery status, and realizes that there is a convenient CP near the business meeting location 30 minutes before it. It sends a notification to Luke who accepts the suggestion and a socket of his "Favourite Vehicle" type is booked in that CP.

2.1.2. Class Diagram

The Class Diagram describes our entire system, composed of eMSP and CPMS. The most important entities described in the diagram are:

User: It represents the user of the eMSP, it has a username, an email, and a password;

Vehicle: Is a vehicle registered by one or more users through a VehicleAPI that verifies the VIN, used as a unique id. The user can set a vehicle as the favourite vehicle for filtering the sockets during StationResearch, and receiving ChargingSuggestion Notifications based on the VehicleBattery and the type of socket supported by the vehicle;

StationResearch: Is based on a Position made by latitude and longitude and a distance range in a certain time frame in order to retrieve the Charging Points in that area;

Reservation: Is made by a user on a specific Socket in a certain time frame at the related tariff or special offer;

ChargingSession: Is related to a Reservation and is described by the amount of energy consumed, the actual battery status of the connected car, the duration of the recharge, and a boolean that recognizes a finished session. When the battery status is full, a ChargingEndedNotification is sent to the user;

Payment: represents a payment, related to a user and a charging session: This class include both successful payments and pending ones if the payment wasn't able to be concluded.

ChargingPoint: It represents a real Charging Point, owned by a CPO. It has a Position (latitude and longitude), different Sockets of different types, having its own status. Each CP has a Tariff for each socket type and could have a SpecialOffers;

EnergySource: It represents an energy source related to a certain CP. It can be a DSOEnergySource, representing a DSO, or a ChargingPointBattery representing a battery connected to the CP. "CurrentEnergySource" is the relation that identifies the resources that can be used at a given time.

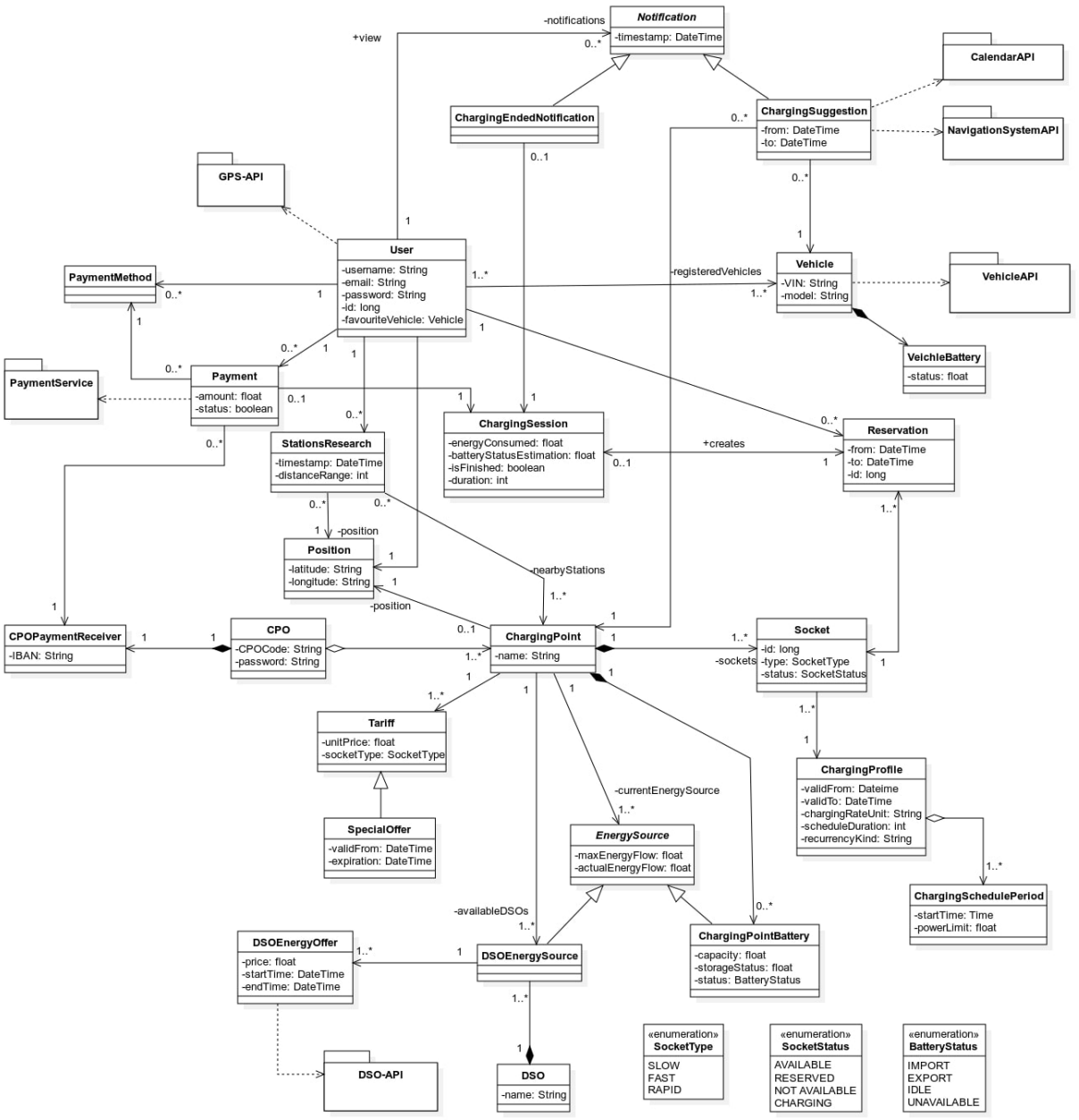

Figure 2.1: Class Diagram

2.1.3. State Diagrams

The following diagrams describe the states that some of the most important entities can assume.

The **Reservation** has four different states. The first one is triggered when a new reservation is created that sets it in the RESERVED state. When the expiration time is passed or the user cancels the reservation will change its state to CANCELLED, ending

the process. If the user doesn't cancel the reservation and it is still valid, it can enter into the CHARGING state until the charging session is over, this will bring to the state COMPLETED, ending the process.

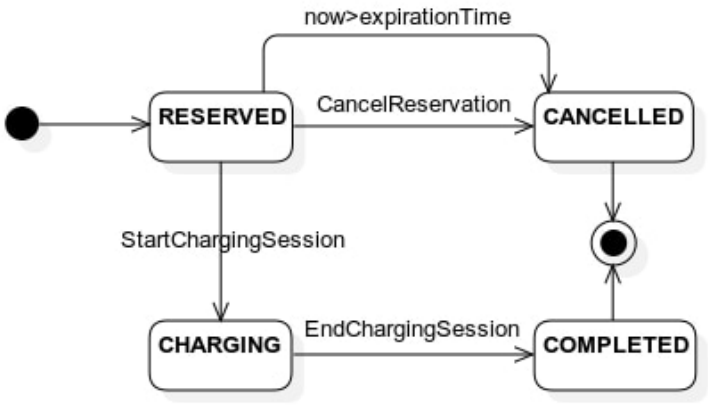

Figure 2.2: Reservation

Each **Charging Session** can take four states. The entry state is the CHARGING one triggered when the user starts the charging session. From this state, we can reach the INVALID state which is useful when an error occurs and we want to avoid billing the user, and the STOPPED state when the user reaches a full charge or wants to stop the charging before it. By the latter, removing the socket from the CP, we enter the ENDED state.

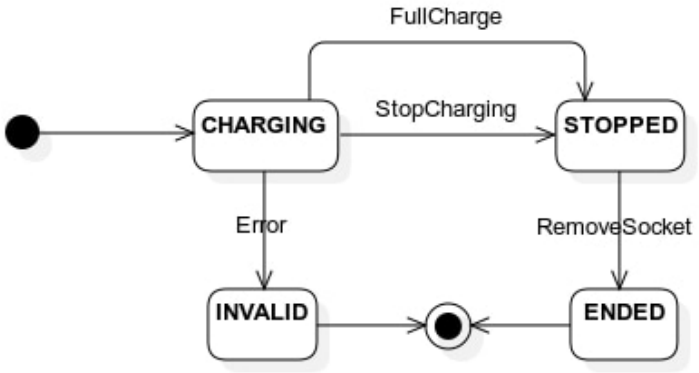

Figure 2.3: Charging Session

The **Sockets** beginning state is the AVAILABLE one, when a socket is reserved it changes its state to RESERVED and can return to AVAILABLE when a reservation is canceled. The CHARGING state can be reached when a charging session is started and returns to AVAILABLE when it's ended. These three states can reach the unavailable state

that is set when an error occurs, from this state we can remove the socket entering the REMOVED state or return to the AVAILABLE one.

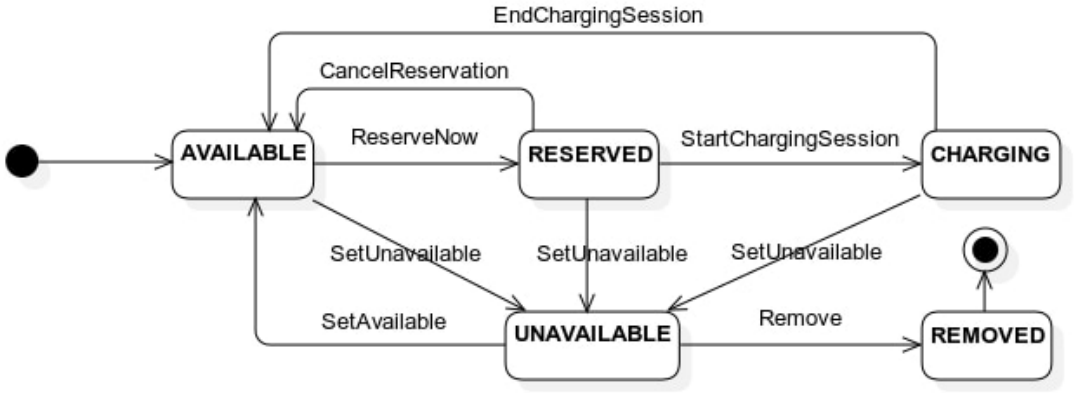

Figure 2.4: Socket

The **Charging Station Battery** has four states. The main state is the IDLE one, which can reach the other three states. When the battery is storing energy it enters the IMPORT state and returns to IDLE when it stops storing or if it is full. When the battery is used as an energy source for a charging point, it enters the EXPORT state and returns to IDLE when it stops charging other vehicles or if it is empty. The last state is UNAVAILABLE which is triggered when an error occurs, from this the battery can return to IDLE or be removed.

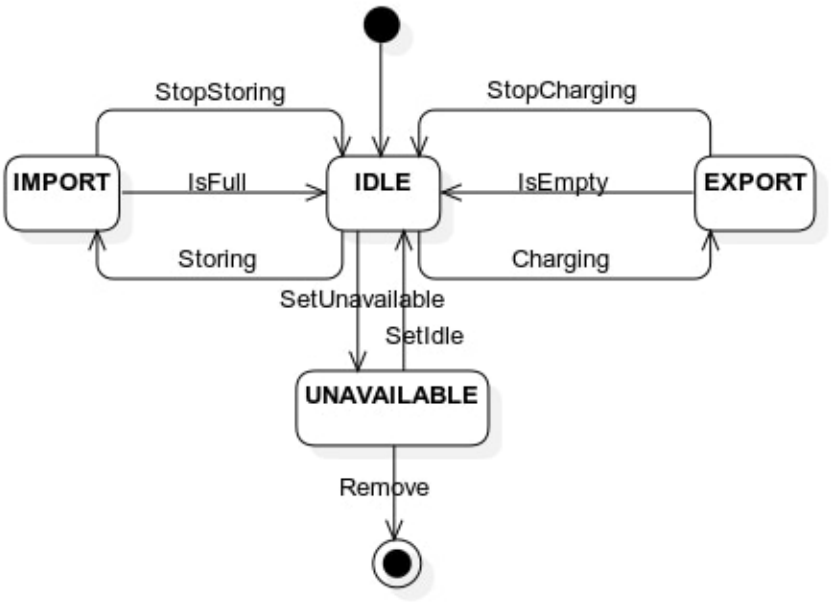

Figure 2.5: Charging Station Battery

2.2. Product functions

2.2.1. eMSP Product Functions

1. **Search and Reservation of a CP.**
   One of the main goals of the eMSP is to allow the User to search the CP around it and reserve a socket if it's free. In order to perform these operations the eMSP needs to have a map view that by default is centered on the User's position and enables it to look at the CP around. Of course, a User can be interested in searching for a CP near a position that has nothing to do with his current position. For this reason, the eMSP has to provide also a way to insert another location and look for the CPs near the position selected. Another important aspect to consider is that the user might be interested in researching a CP that has a free socket or that has a compatible socket with his vehicle. The User should have the possibility to insert vehicle information into the application and the system can use the user's favourite vehicle to filter the results on the map. In order to achieve maximum flexibility we designed the system in a way that after the research of a CP, all kinds of CPs are displayed (even the ones that have all the sockets currently busy or incompatible), but they will be shown to the user in different ways depending on which of these three categories they belong:

   (a) The CP has one or more sockets that are compatible and currently available. The user is very interested in these CPs since are the ones where he can reserve a socket compatible with his favourite vehicle;

   (b) The CP has one or more sockets that are compatible with his favourite vehicle, but they are all busy. The user can be interested in this kind of CP since he may be interested in waiting until a socket is free;

   (c) The CP has all sockets that are not compatible with the user's favourite vehicle. This case can be useful for users that have multiple vehicles.

   The user of course can reserve a free socket for a charging session. The reservation process requires that the user has no pending payments. When a reservation is made the User has 20 minutes to go to the CP and start the charging process. The User can also delete a reservation if has changed his mind, while if he doesn't start the charging process within 20 minutes the reservation will expire. The system is designed in a way that each user can have at most one reservation active at a time.

2. **Charging session and payment.**
   Another fundamental functionality offered by the eMSP is the possibility to start a recharge, monitor it, and pay for the service. The user in eMSP has to reserve a socket before starting the charging process. So when the drivers arrive at the CP can open the eMSP application and select the "start charging process" option, at this point, the CP's socket connector is unlocked and the user can insert the socket from his car into the CP's connector. The CP is able to detect that a car is plugged in and can start the energy transfer while locking the socket. During the charging process, the eMSP gives the user the possibility to observe the charging process and the information about his vehicle's battery. The CP blocks the charging process in an automatic way the battery is full and in this case the eMSP notifies the user through a pop-up notification that the process is completed. The system gives also the user the possibility to stop a charging process when he wants before the battery of his vehicle is full. After a charging process is stopped the user can unlock the socket; the charging session is ended now. The system at this point asks the user to pay for the charging process. When the user allows the eMSP to process the payment, the eMSP use the payment method selected by the user to pay through an external payment service. If the payment fails, it will be marked as unsolved. The system will prevent the user with an unsolved payment to make new reservations and will give them the possibility to pay at a later time. The system notifies the user of the payment's success or failure.

3. **Suggestions.**
   A very important part that characterizes our eMSP is the feature of making suggestions to the user and proposing a socket to reserve. When a user logs in for the first time into the eMSP, the eMSP will ask it permission to access his calendar and navigation system to retrieve information to optimize suggestions. The eMSP will interact with these two applications of the user but also with an external API that retrieves the battery status of the favourite vehicle. The eMSP will make a suggestion when the vehicle battery reaches a certain percentage. When this happens the eMSP will interact at first with the navigation system to retrieve the navigation path if inserted. In this case will compute the best stop option to minimize the price of the recharge and the user's time cost to reach the CP and complete the recharge, considering the information of his favourite vehicle and the status of the socket of each CPs under a certain range. Then the system will interact with the calendar system in order to understand if the user has an appointment schedule during the

day and acquire the relative location. The eMSP will combine all this information to make a suggestion. If the calendar and navigation system do not give relevant information about the user the system will use user's the current position to make a suggestion. The user will receive the suggestion in form of a pop-up notification and can accept or decline it. When a suggestion is accepted the system will reserve the relative socket. If an error occurs during this process, the eMPS will recompute another suggestion for the User.

2.2.2. CPMS product functions

1. **Manage energy source.**
   The CPMS offers the CPO several functionalities, one of the most important is the energy source acquisition system. The CPMS is designed in a way that is able to minimize the cost that the CPO has to bear to provide energy to the CPs. For this purpose, the CMPS has two different "abilities" that combined enable the system to behave in the optimal way to achieve cost efficiency.

   Firstly the system is able to select from which DSO acquires energy in an optimal way, making the choice in a different way for each CP managed, considering the expected traffic, energy prices, and DSO's availability and offers for each CP.

   Another point is to exploit the battery usage for each CP at its best, computing the optimal strategy to schedule the energy acquisition period vs exploiting the battery.

   These two aspects combined let the CPMS be maximum efficient. By the way, the CPMS allows the CPOs to control directly both aspects (Manual Mode). In fact, the CPO can manually select from which DSO acquires energy and how to use the battery for each CP. In addition, the CPO can select the Manual Mode for one of the two functionalities and let the CPMS automatically optimise the other one. For example, the CPO can force the selection of the DSO and then let the CPMS decide how to use the energy provided by that DSO. By default, the CPMS is set to Automatic Mode for both functions and the CPO can choose to select Manual Mode for just some of the charging points that he owns.

2. **Manage sockets.**
   The CPMS offers the CPO the possibility to manage the sockets' status for each CP. This means that the CPMS gives the CPO the possibility to change the socket availability, making it unavailable or available for reservation. The CPMS of course enables the CPO to add or remove CP and register or delete sockets.

Each socket has a maximum power that can provide to a plugged car that depends on the socket type. By the way, it can be possible that in some circumstances the amount of energy that a certain CP can provide to its socket, is not sufficient to guarantee that all the sockets are able to work at that level at the same time.

For this reason, every time the CPMS changes the DSO offer of energy, consider the minimum amount of energy that it guarantees and based on this compute the minimum level of energy that each socket can guarantee to a plugged car, considering also the scenario in which all sockets are plugged to a car. The system can also consider excluding one or more sockets to guarantee an adequate minimum level of energy for the others. Anyway, considering the current energy power provided by the DSO and the number of cars plugged into the system try to reach an optimal behaviour giving power to each socket up to its maximum power.

3. **Offers and Tariff.**
   The CPMS offers CPOs the possibility to publish the tariff for each CP on the system (Manual Mode). The user of the eMSP will pay his recharge according to the Tariff valid when the reservation was made. The CPMS also gives the possibility to create a special offer, that consists of a change of price for one or more CPs for a limited amount of time. The eMSP will display the special offers in a dedicated part giving particular visibility to them. The CPO can also decide to let the CPMS set tariffs and special offers according to the prices of the DSOs and the availability of the batteries (Automatic Mode). This decision can be made for all charging points or for just some of the ones.

2.3. User characteristics

1. **User**
   As users, we intend the user of the eMSP. This user can register to the system through a valid email and a password. At the moment of registration, the eMSP will send an email to verify the user.

2. **CPO**
   The CPO is the actor that uses the CPMS and manages CPs. Since the CMPS is not freely distributed, an API has to verify and validate the code provided by the

CPO to register and log in to the system.

2.4. Assumptions, dependencies and constraints

D1 The sockets have their own sensor that is able to estimate the status of the battery of the vehicle that is connected.

D2 The user can always physically access the socket in the reserved time slot.

D3 There exists a sensor of the CP that is able to monitor the quantity of energy that the DSO is currently providing.

D4 There exists an external service that checks the validity of billing information.

D5 There exists an external service that checks the validity of the VIN code of the vehicles.

D6 There exists an external service that retrieves the status of the battery of a vehicle when a socket is not plugged.

D7 All the sockets of a certain type in the same charging point have the same tariff.

D8 A CPO is identified by a unique identifier provided by the local authorities.

D9 A CP is identified by a unique identifier provided by the local authorities.

D10 The CP is able to detect when a user plugs/unplugs the socket.

D11 Each CP can have at most one DSO that provides the energy at a time.

D12 A reservation is valid from the moment on which is made to the following 20 minutes, after which it is cancelled.

D13 A charging session can start only from an existing valid reservation.

D14 For each charging point there exists at least one DSO that can provide the energy to it.

D15 The CP automatically stops the energy flow when it detects that the battery of the car is full.

3 | Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

eMSP

The eMSP must be a mobile application available for every kind of user (both for IOS and Android OS). It also must be easy to use and intuitive, and for this reason, is important to take into account some crucial and specific aspects. Firstly the map view is a very important aspect and the map should be on the user's main page. Different categories of CPs must be distinguished in a very clear and easy-to-understand way, choosing for example to represent in the map the CPs with icons with different shapes and colors. The zoom level of the map when a user researches a location or goes to the main page is important too, cause it represents the range of research of the CPs. Another important aspect of the eMSP user interface is the notification message. In fact, we can suppose that many times the system will send a pop-up notification with a suggestion while the user is driving, maybe with his navigation system activated. For this reason, when a pop notification is open the application should display a very simple page, from which the user can accept the suggestion and reserve the suggested socket by pressing only one big visible button on the screen. This is important in order to avoid distracting the user while driving. The page used to monitor the charging process should include all important information for the user, such as the current battery level and the estimated time to complete the charge.

CPMS

The CPMS must be easily accessible from the web. The interface should allow the CPOs to easily manage all of their CPs and the cognitive complexity required to visualize data relating to CPs' internal and external status, and DSOs' offer must be minimized. For this reason, is important to display all the most important offers information on a single

page that can be scrolled, and this preview should include aspects of the offer that CPOs usually compare to make decisions, such as price, the date until the offer is valid and the maximum level of energy offered. A filter should be available on the DSOs offers page to enable the CPO to search for offers of a certain kind (minimum date of expiration, minimum energy level offered, ...), but also it should be possible to order offers according to relevant dimensions (price, DSO provider, amount of energy, ...).

3.1.2. Hardware Interfaces

eMSP

The user needs a smartphone with an IOS or Android OS installed and an Internet Connection. Also, GPS should work to use all application functionalities.

CPMS

The CPO needs a device with an internet connection and a Browser to access the web App.

3.1.3. Software Interfaces

We designed the system in order to comply with standards in the sector. This is very important since the eMSP designed should be able to interact with all CPMSs that complain with the standards and also the CPMS we designed is able to interact with external eMSP. The CPMS should also use a standard protocol to communicate with the CPs to operate on sockets and retrieve relative information and communicate with the DSOs to retrieve offers. The eMSP must be also able to use the User's Calendar, Navigation system, and GPS but also access the vehicle's battery and other information through APIs. Also, the payment process is external to the system and requires an apposite API interaction.

More in detail:

- **OCPI** standard is used to regulate communication among eMSP and CPMS. It has been chosen since is the standard provided by the industry, provides a wide range of operations, and enhances the flexibility of the system;
- **OSCP** is the standard chosen for the communication between DSO and CPMS, it's the industry standard;
- **OpenADR** is the standard chosen for the communication of the prices between DSO and CPMS;

- **OCPP** is the standard chosen for the communication between CPMS and CP;
- **Vehicle API** is the protocol chosen for the interaction with the vehicles' batteries. This was chosen since provide since is compatible with a wide range of automotive manufacturers and provides a wide range of information including the battery level.

  There are no particular constraints regarding the interaction between GPS, the Navigation System, and the Calendar, by the way, the system should be able to interact with the more popular application of these types like GoogleCalendar, Calendar(iOS), GoogleMaps, Apple Maps. Also for the payment system, there are no particular constraints.

3.2. Functional Requirements

3.2.1. User Use Cases

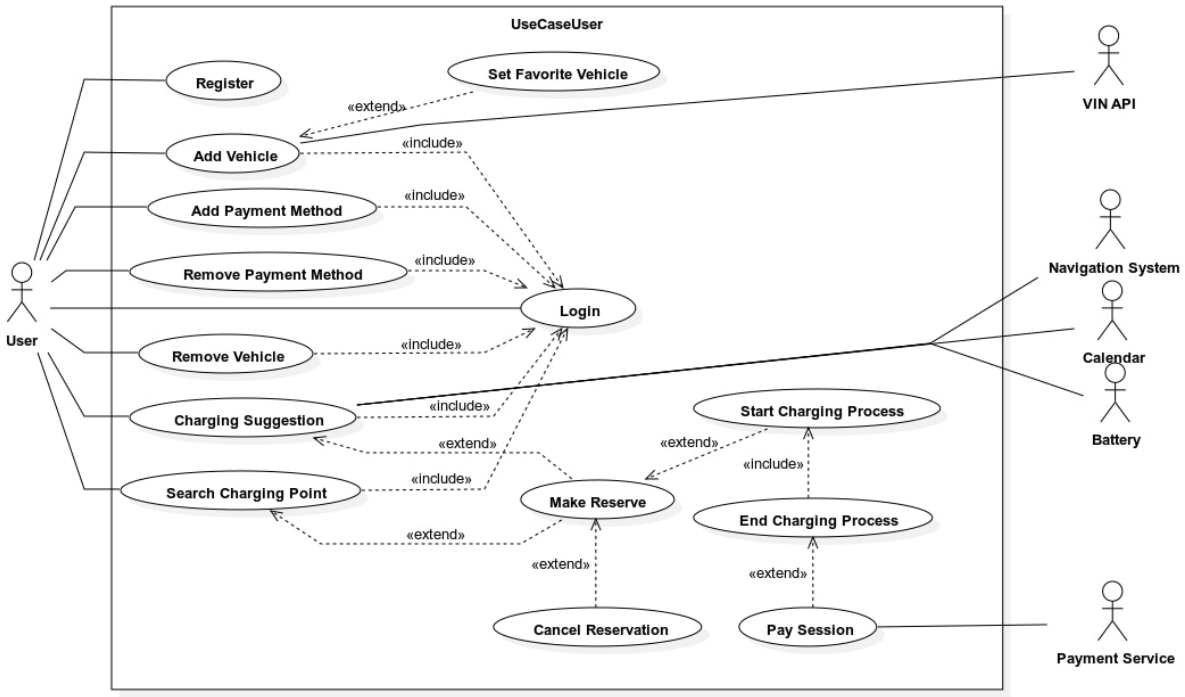

Figure 3.1: User Use Cases

ID	**UC1**
Name	**User Registration**
Actor	User
Entry Condition	The user does not have an account and is on the login view
Event Flow	1. The user presses the "Register account" button; 2. The page with the registration form shows up; 3. The user compiles the required fields(email, password, confirm password, name, surname); 4. The eMSP validates the data; 5. The eMSP sends a confirmation email with a code; 6. The user inserts the code of the email in the form; 7. The eMSP creates the user; 8. The eMSP displays a success message.
Exit Conditions	The user account is created
Exceptions	1. The user does not enter all mandatory data; 2. The user data aren't valid; 3. The user already exists; 4. The user does not enter the correct confirmation code. • In all cases the eMSP will notify the user.

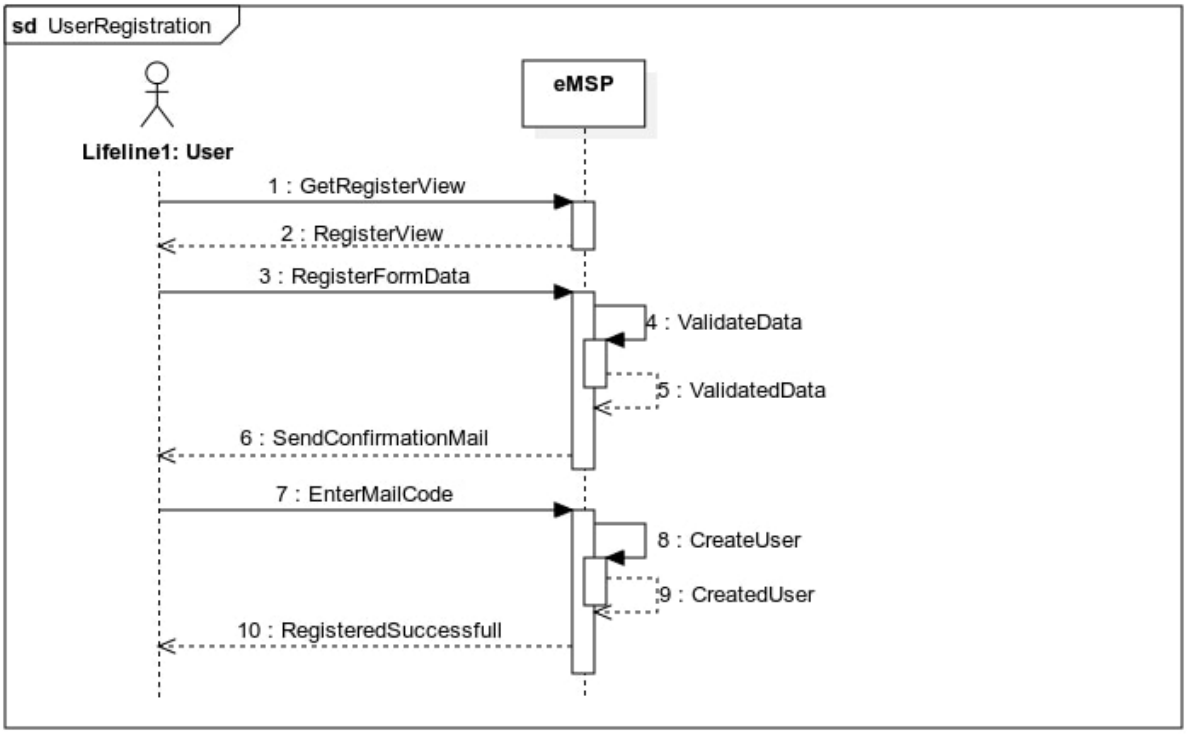

Figure 3.2: User Registration

ID	UC2
Name	**User Login**
Actor	User
Entry Condition	The user is registered but not logged in and is on the login view
Event Flow	1. The user inserts the combination of his email and password; 2. The eMSP processes the information; 3. The eMSP displays a success message.
Exit Conditions	The user is logged into the eMSP
Exceptions	1. The user does not enter his credentials before submitting; 2. The user enters invalid credentials. • In all cases the eMSP will notify the user.

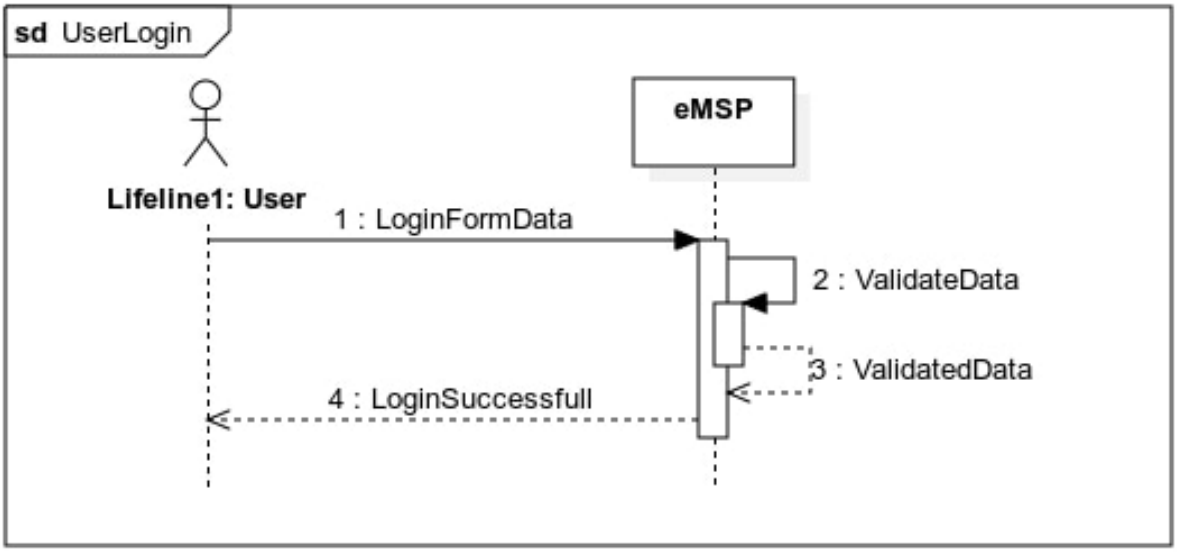

Figure 3.3: User Login

ID	UC3
Name	**Add Vehicle**
Actor	User, VIN API
Entry Condition	The user is on the main menu of the application page
Event Flow	1. The user selects the overview of the vehicle; 2. The eMSP retrieves a list of all vehicles provided by the user; 3. The user selects the “add new vehicle” option; 4. The eMSP displays a form in which the user can select the car to insert providing the VIN code; 5. The user inserts the VIN code of his car; 6. The eMSP checks the VIN validity by sending a message to the VIN API; 7. The VIN API sends a response to the eMSP; 8. The eMSP shows a summary of the information regarding his car and asks for confirmation; 9. The user chooses the confirm option; 10. The eMSP displays a success message; 11. The user set it as "favourite vehicle".
Exit Conditions	A new vehicle is inserted as favourite
Exceptions	The VIN code provided is not correct. • The eMSP asks the user to insert the correct code.

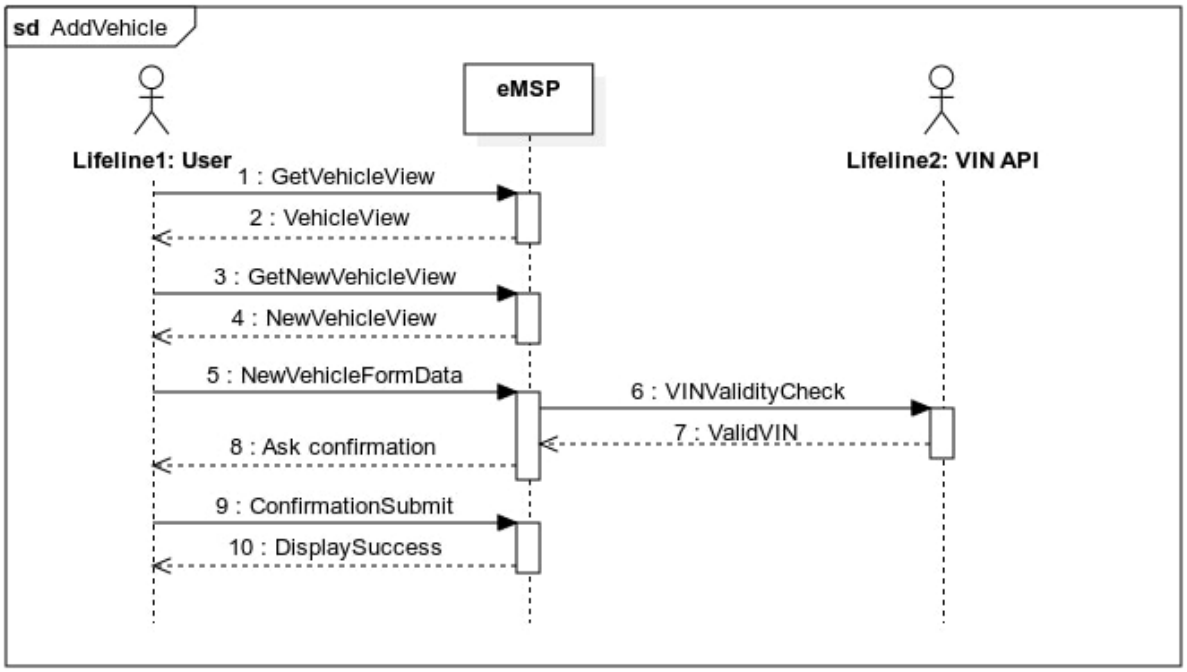

Figure 3.4: Add Vehicle

ID	UC4
Name	**Set Favourite Vehicle**
Actor	User
Entry Condition	The user is on the main menu of the application
Event Flow	1. The user selects the vehicle overview; 2. The eMSP retrieves a list of all vehicles provided by the user; 3. The user selects a vehicle; 4. The eMSP displays a summary of all vehicle information; 5. The user selects the "set as Favourite" option; 6. The eMSP display a success message.
Exit Conditions	The vehicle is set as favourite from the eMSP
Exceptions	The vehicle doesn't exists

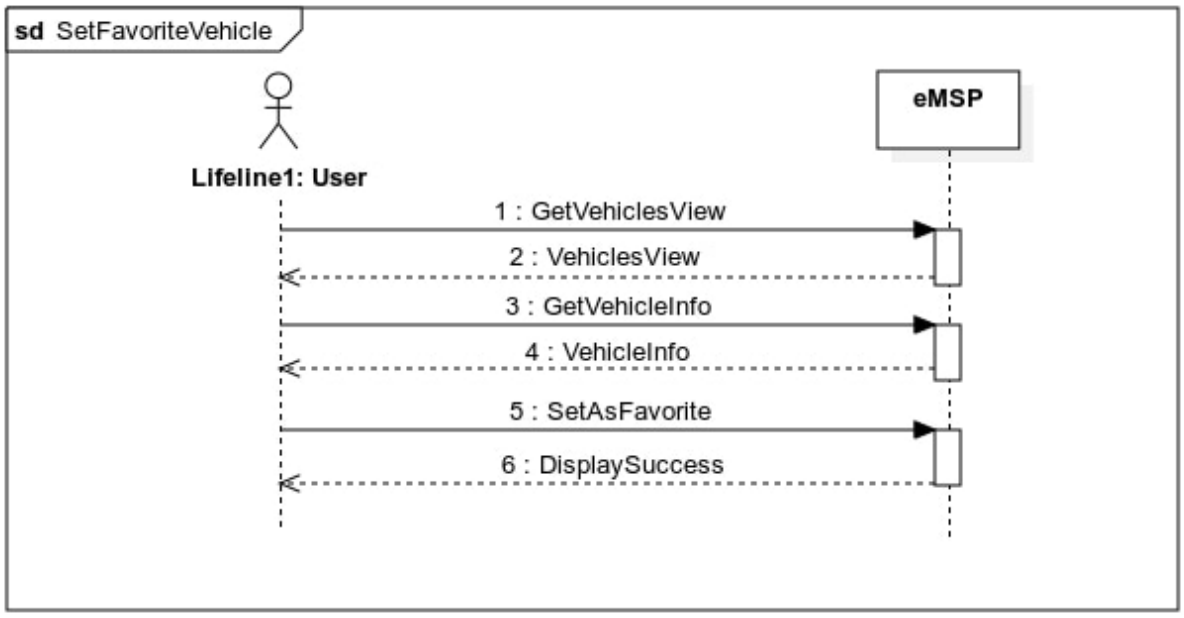

Figure 3.5: Set Favourite Vehicle

ID	UC5
Name	**Remove Vehicle**
Actor	User
Entry Condition	The user is on the main menu of the application
Event Flow	1. The user selects the vehicle overview; 2. The eMSP retrieves a list of all vehicles provided by the user; 3. The user selects a vehicle; 4. The eMSP displays a summary of all vehicle information; 5. The user selects the "remove vehicle" option; 6. The eMSP asks for confirmation; 7. The user confirms the action; 8. The eMSP display a success message.
Exit Conditions	The vehicle is removed from the eMSP
Exceptions	The vehicle doesn't exists

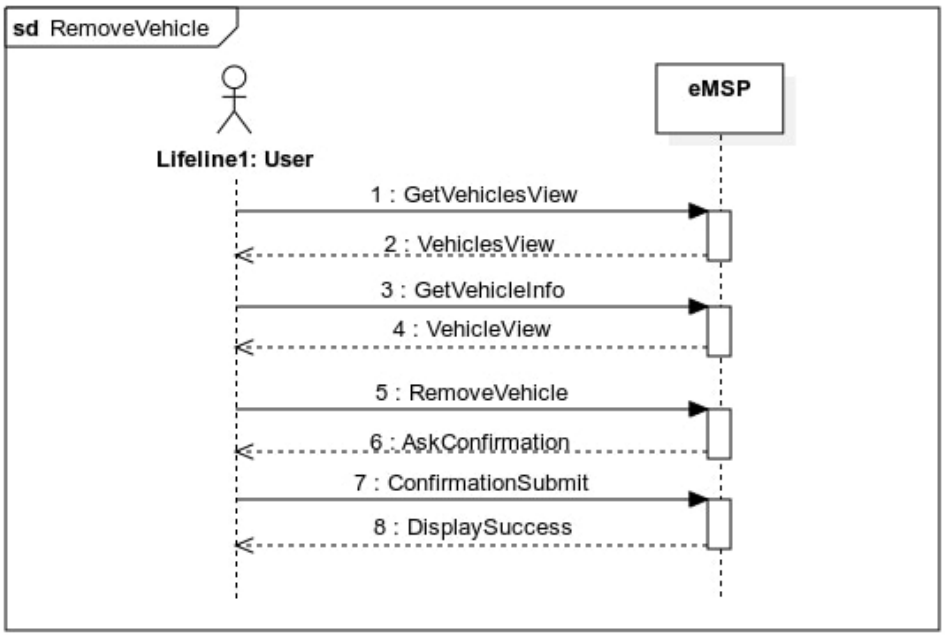

Figure 3.6: Remove Vehicle

ID	UC6
Name	**Add Payment Method**
Actor	User, Payment Service
Entry Condition	The user is logged in and in the payment methods page
Event Flow	1. The user clicks the "Add Payment Method" button; 2. The eMSP shows the add payment method form; 3. The user sends the payment information to the eMSP; 4. The eMSP sends the information to the payment service for validation; 5. The payment service responds with a valid message; 6. The eMSP display a success message.
Exit Conditions	A new Payment Method is created
Exceptions	The payment method isn't valid. In this case, the eMSP will notify the user.

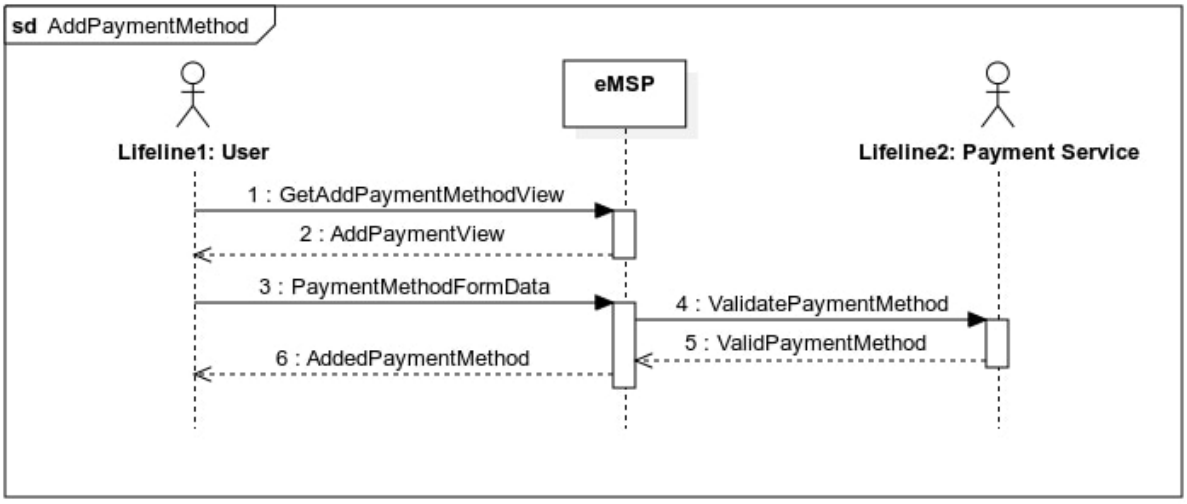

Figure 3.7: Add Payment Method

ID	UC7
Name	**Remove Payment Method**
Actor	User
Entry Condition	The user is logged in and in the payment methods page
Event Flow	1. The user clicks on the payment method that he wants to delete; 2. The eMSP displays the payment method information; 3. The user clicks on the "Delete" button; 4. The system asks for confirmation; 5. The eMSP display a success message.
Exit Conditions	The payment method is removed
Exceptions	The payment method doesn't exist

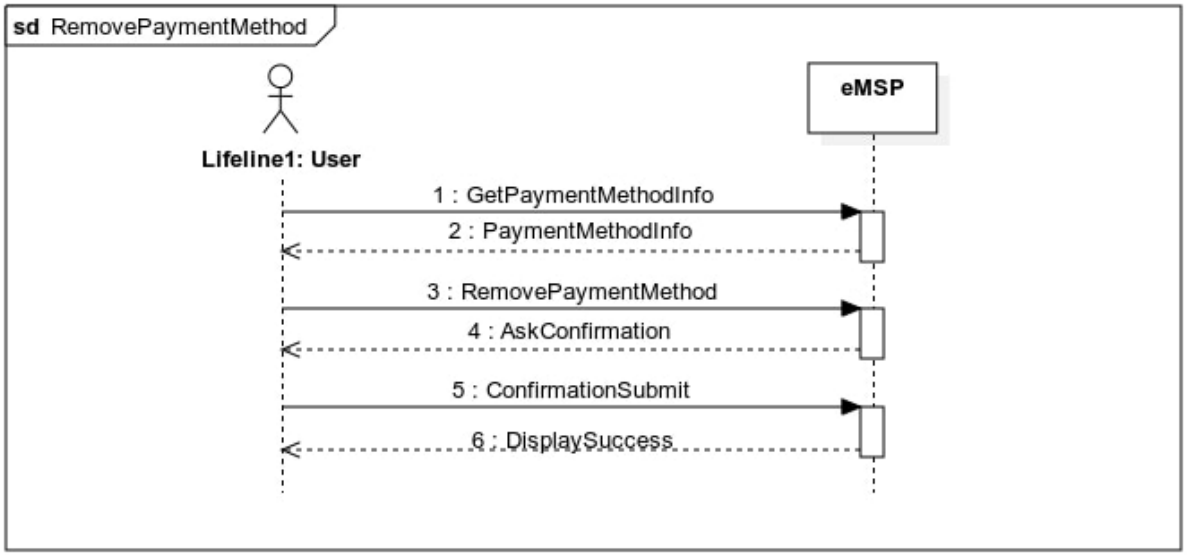

Figure 3.8: Remove Payment Method

ID	UC8
Name	**Make Reservation**
Actor	User, CP
Entry Condition	The user is logged in and is on the map page
Event Flow	1. The user requests CP info in an area by moving into the map view; 2. The eMSP displays in the map portion, the CP distinguish by three cases: (a) At least a free socket is compatible with the favourite vehicle; (b) All sockets compatible with the favourite vehicle are busy; (c) All sockets are incompatible with the favourite vehicle. 3. The user clicks on one of the visualized stations; 4. The eMSP sends a CP's info request to the CPMS; 5. The CPMS sends the CP info to the eMSP; 6. The eMSP displays the station information; 7. The user selects a free socket; 8. The eMSP displays socket information; 9. The user clicks the reserve button; 10. The eMSP checks that the user doesn't have not paid sessions or active reservations; 11. The eMSP sends the reserve request to the CPMS; 12. The CPMS sends the reserve to the CP; 13. The CP sends the response to the CPMS; 14. The CPMS sends the response to the eMSP; 15. The eMSP displays a success message.
Exit Conditions	The user reservation is created
Exceptions	1. The user has an old not paid session; 2. The user already has an active reservation; 3. The selected socket is no more available. • In all cases the eMSP will notify the user and prevent the reservation.

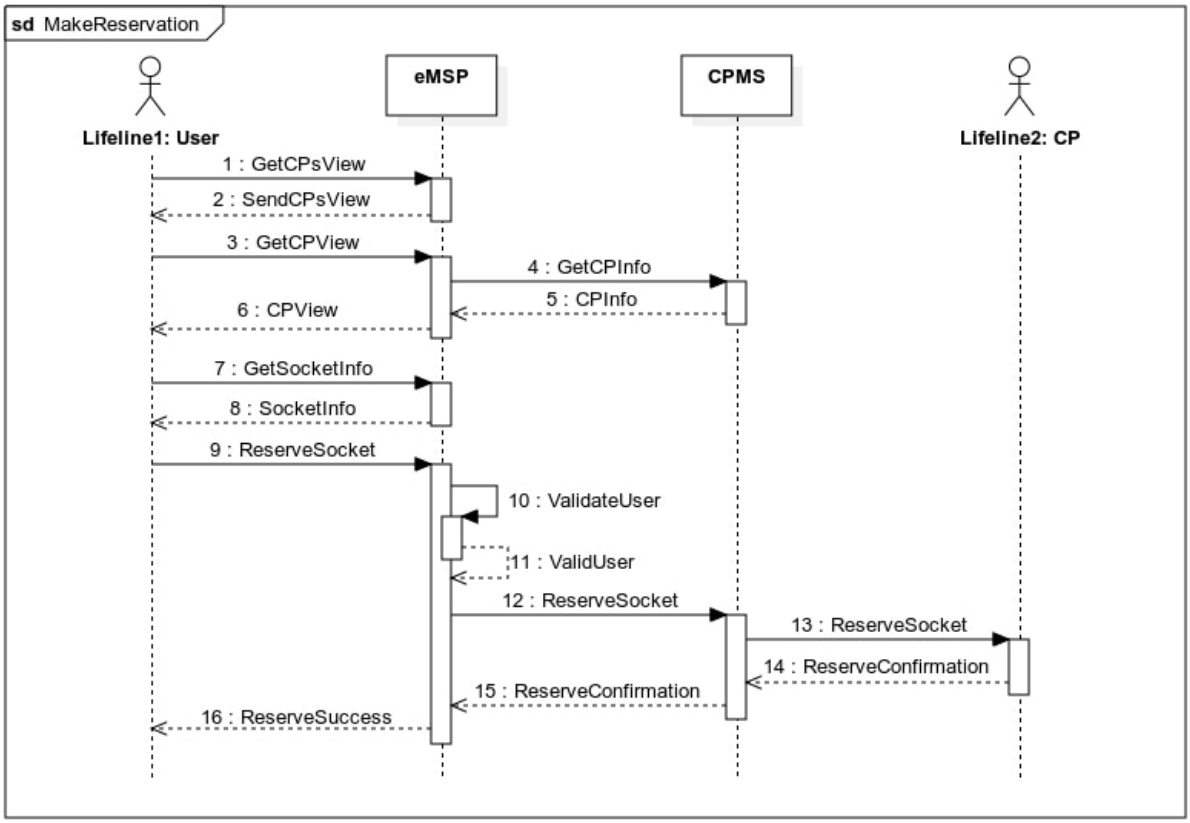

Figure 3.9: Make reservation

ID	UC9
Name	**Manage Charging Suggestion**
Actor	User, Navigation System, Calendar, Battery, CP
Entry Condition	The user favourite vehicle has a low battery
Event Flow	1. The eMSP retrieves the favourite vehicle's battery status through API; 2. The eMSP retrieves the position from the GPS of the user; 3. The eMSP retrieves the schedule of the user through the calendar; 4. The eMSP retrieves the current navigation path, if present, of the user through its Navigation System; 5. The eMSP computes the best set of CP to charge the Vehicle; 6. The eMSP sends a notification to the User including the resume of the suggested stops; 7. The User opens the application from the popup notification; 8. The eMSP display the charging suggestion resume; 9. The User accepts the charging suggestion; 10. The eMSP checks that the user doesn't have paid sessions; 11. The eMSP sends the reserve request to the CPMS; 12. The CPMS sends the reserve to the CP; 13. The CP sends the response to the CPMS; 14. The CPMS sends the response to the eMSP; 15. The eMSP displays a success message.
Exit Conditions	The eMSP completes the reservation process
Exceptions	1. The suggestion is declined; 2. The reservation fails because the sockets are busy. • In this case the eMSP calculates a new suggestion.

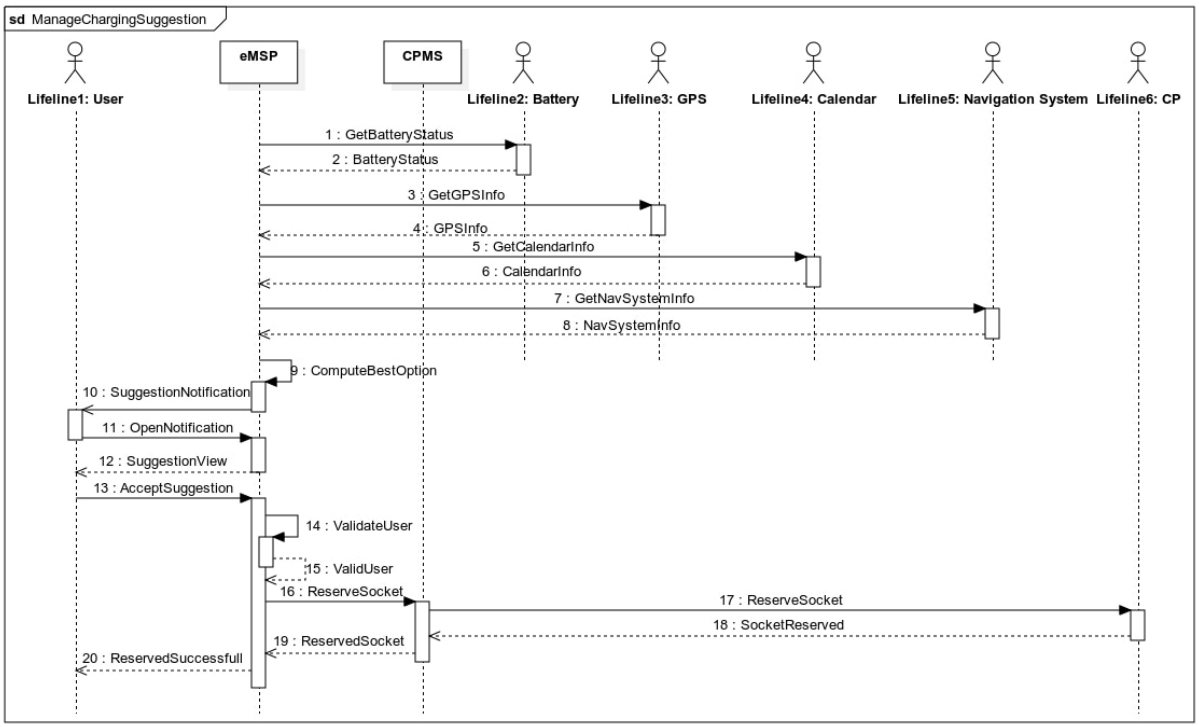

Figure 3.10: Manage charging suggestions

ID	UC10
Name	Cancel Reservation
Actor	User, CP
Entry Condition	The user is logged in and is in the reservations tab
Event Flow	1. The user clicks on the current reservation; 2. The user visualizes the current reservation information; 3. The user clicks on the "Cancel Reservation" button; 4. The eMSP sends the cancellation request to the CPMS; 5. The CPMS sends the cancellation to the CP; 6. The CP sends the response to the CPMS; 7. The CPMS sends the response to the eMSP; 8. The eMSP displays a success message.
Exit Conditions	The reservation status is updated to cancelled and the socket is available
Exceptions	The reservation time is expired. • In this case the system will notify the user.

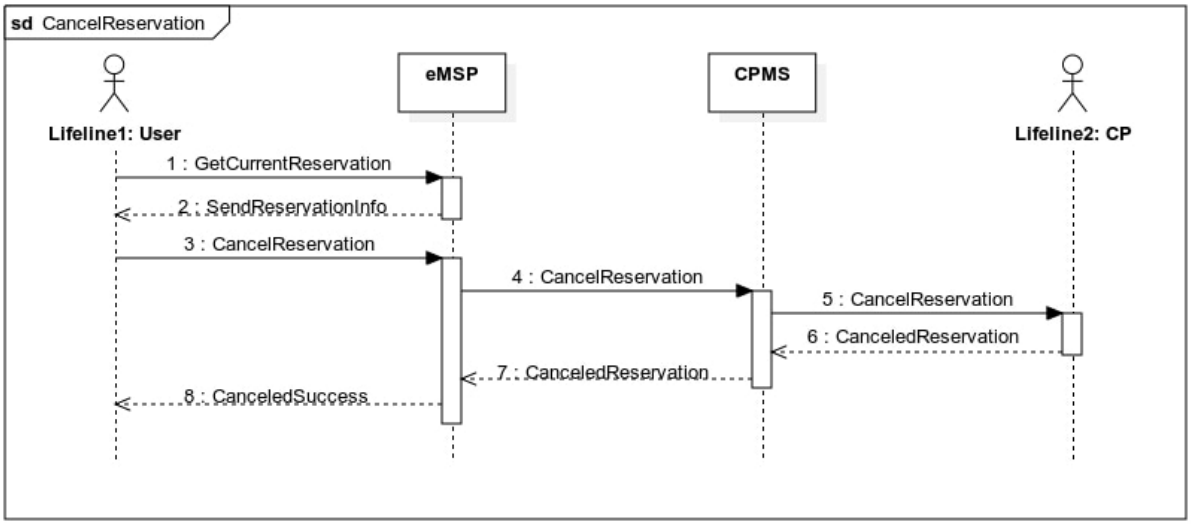

Figure 3.11: Cancel reservation

ID	UC11
Name	**Start Charging Process**
Actor	User, CP
Entry Condition	The user is logged in and in the current reservation tab
Event Flow	1. The user clicks the "Start Session" button; 2. The eMSP sends the start session request to the CPMS; 3. The CPMS sends the start session request to the CP; 4. The CP accepts the command and unlocks the socket; 5. The CP waits for the user to plug the socket; 6. The CP recognizes a plugged socket; 7. The CP locks the socket; 8. The CP starts the energy flow to the vehicle; 9. The CP informs the CPMS that the CPMS that the energy flow is started; 10. The CPMS sends to the eMSP the information about the started session; 11. The eMSP sends a notification to the user to inform him about the start of the charging session.
Exit Conditions	The charging process is started
Exceptions	1. The CP can't unlock the socket; 2. The CP can't start the charging process; 3. If the user doesn't plug the socket after a certain timeout, the CPMS will invalidate the session. • In all cases the eMSP will notify the user.

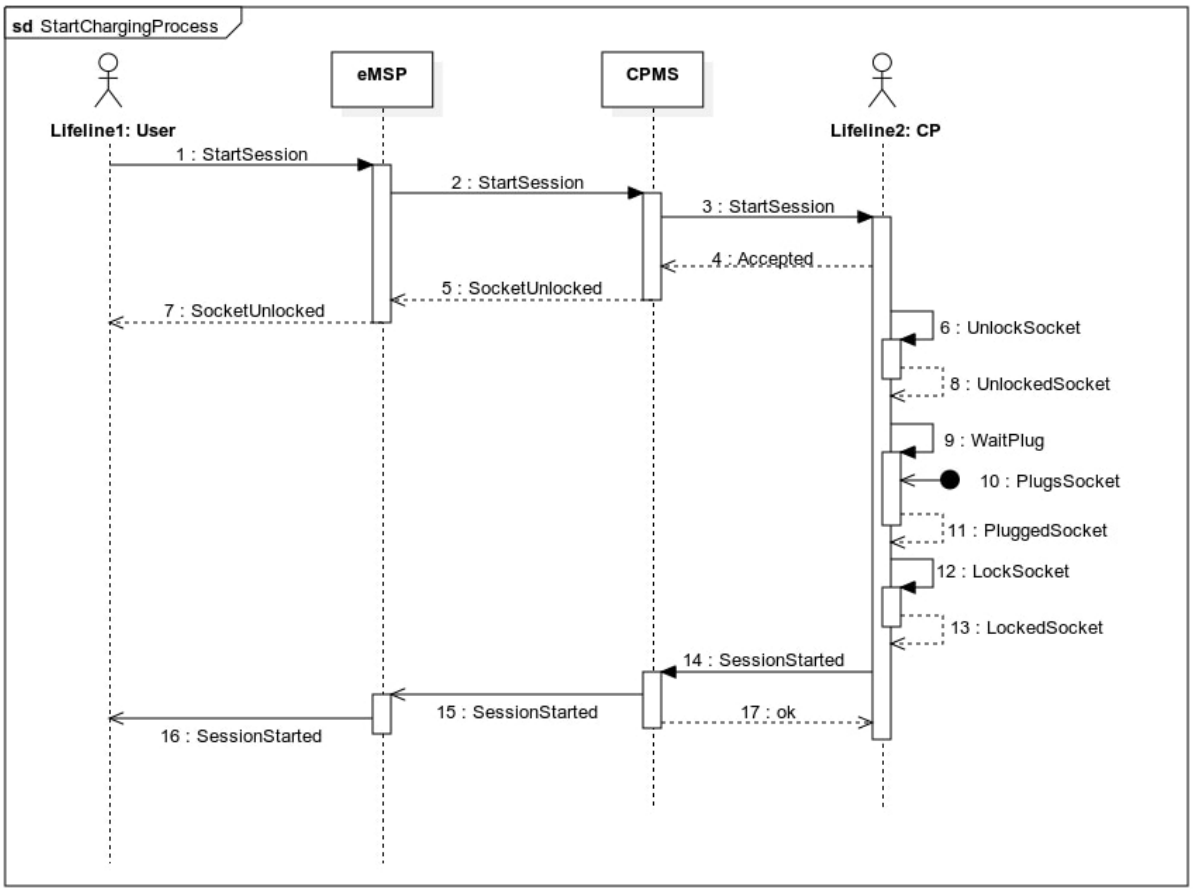

Figure 3.12: Start charging process

ID	UC12
Name	**End Charging Process**
Actor	User, CP
Entry Condition	The user has an active charging session, is logged on and • the battery of his car is full and he has received a notification for it, or • wants to stop the energy flow before the battery is full
Event Flow	1. The user clicks on the current charging process button; 2. The eMSP sends the charging session request to CPMS; 3. The CPMS sends the charging session request to the CP; 4. The CP sends the charging session to the CPMS; 5. The CPMS sends the charging session to the eMSP; 6. The eMSP displays the charging session to the user; 7. The user clicks on the "Stop Session" button; 8. The eMSP sends the stop session request to the CPMS; 9. The CPMS sends the stop session request to the CP; 10. The CP acknowledges the request to the CPMS to inform him that the request will be executed; 11. The CPMS notifies the eMSP that the session will stop; 12. The eMSP notifies the user; 13. The CP stops the energy flow and unlocks the socket; 14. The CP waits for the user to unplug the socket; 15. The CP recognizes that the socket is unplugged; 16. The CP sends an update to the CPMS about the new status of the socket; 17. The CPMS sends a confirmation message to the CP.
Exit Conditions	The charging process is ended
Exceptions	1. The charging process is already ended; 2. The CP can't stop the charging process. • In all cases the system will notify the user.

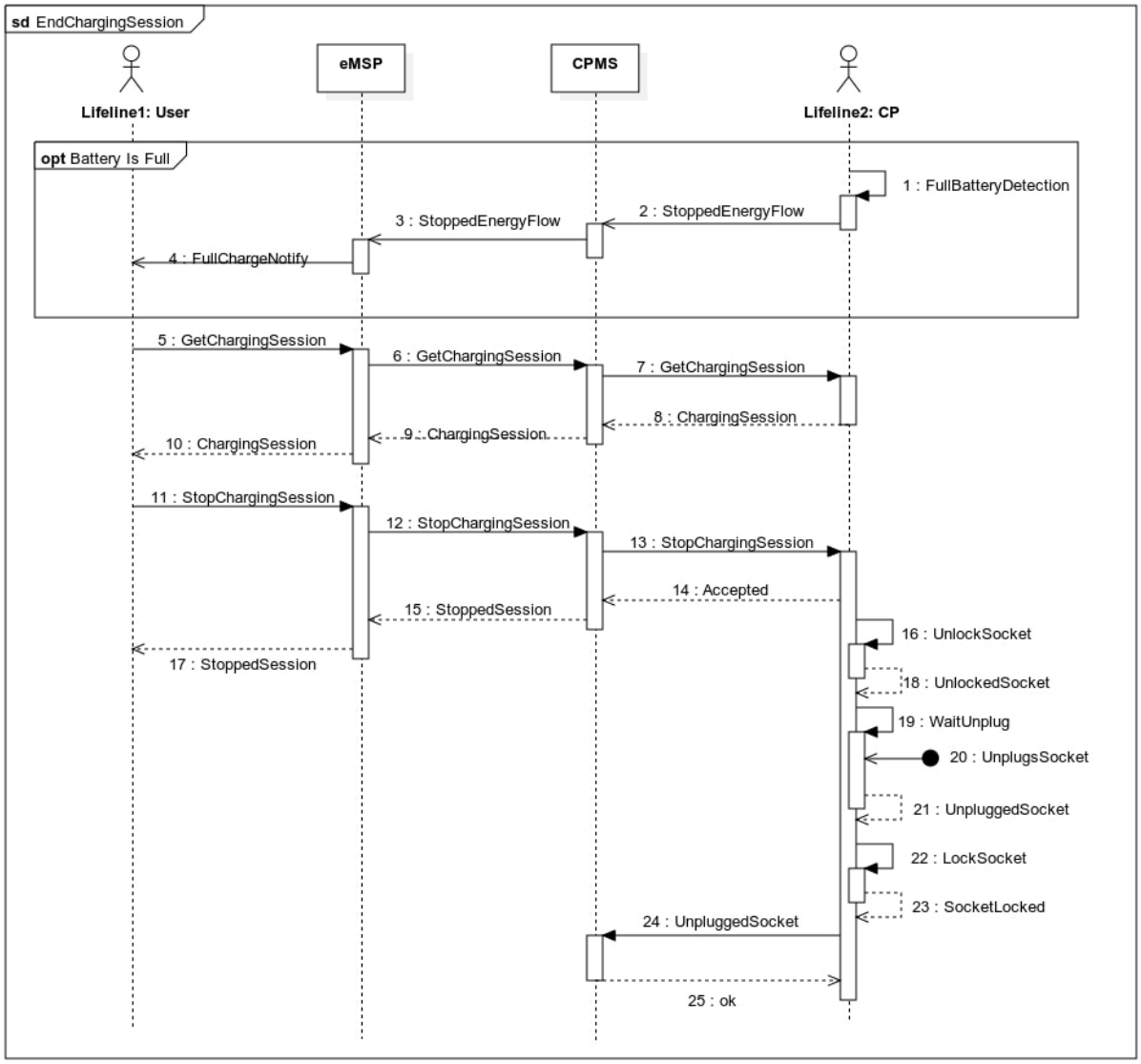

Figure 3.13: End charging process

ID	UC13
Name	**Pay Session**
Actor	User, Payment service
Entry Condition	1. The user is logged in and just ended the charging process; 2. The user is logged in and clicks on pay not paid charging session.
Event Flow	1. The user selects one of its payment methods; 2. The eMSP retrieves the user's billing information; 3. The eMSP retrieves the CPO's billing information; 4. The eMSP sends the information to the external payment service; 5. The eMSP gets the response from the external payment service; 6. The eMSP updates the payment status of the charging process.
Exit Conditions	The charging process is paid
Exceptions	1. The payment method is not valid; 2. The external payment service returns an error. • The eMSP adds the session to the not paid sessions.

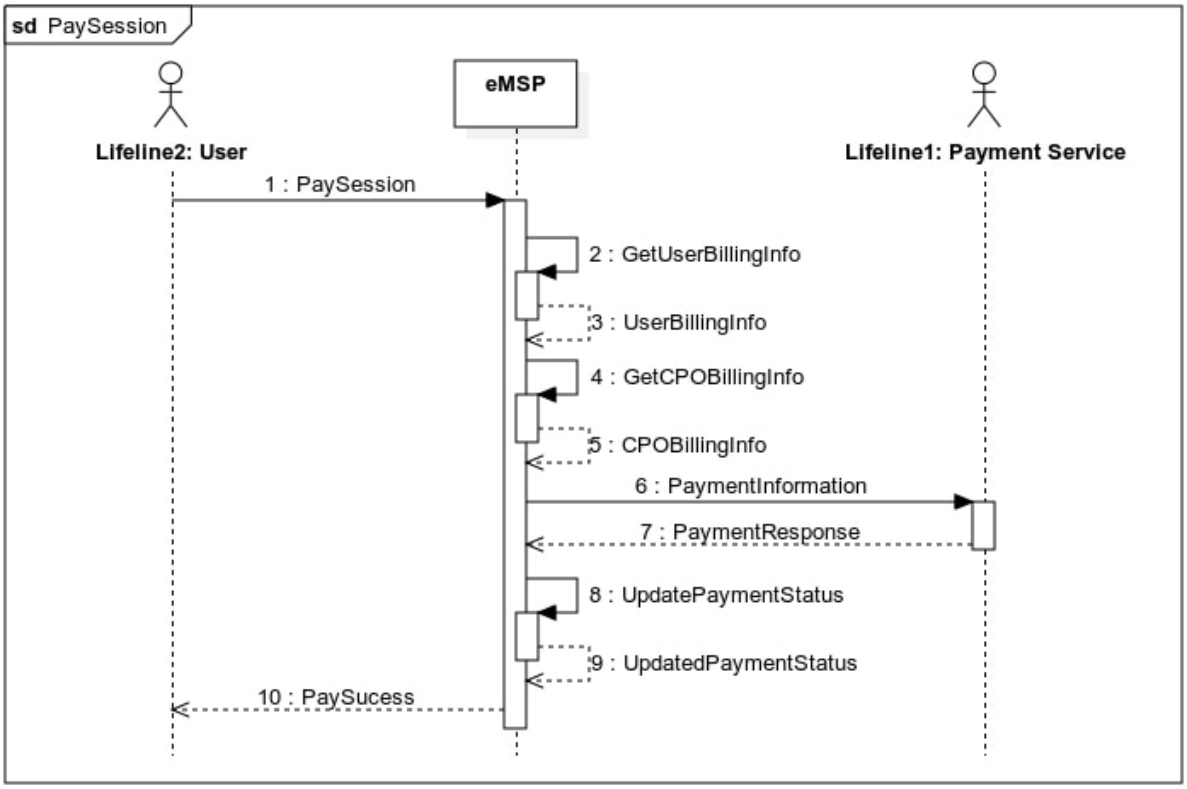

Figure 3.14: Pay Session

3.2.2. CPO Use Cases

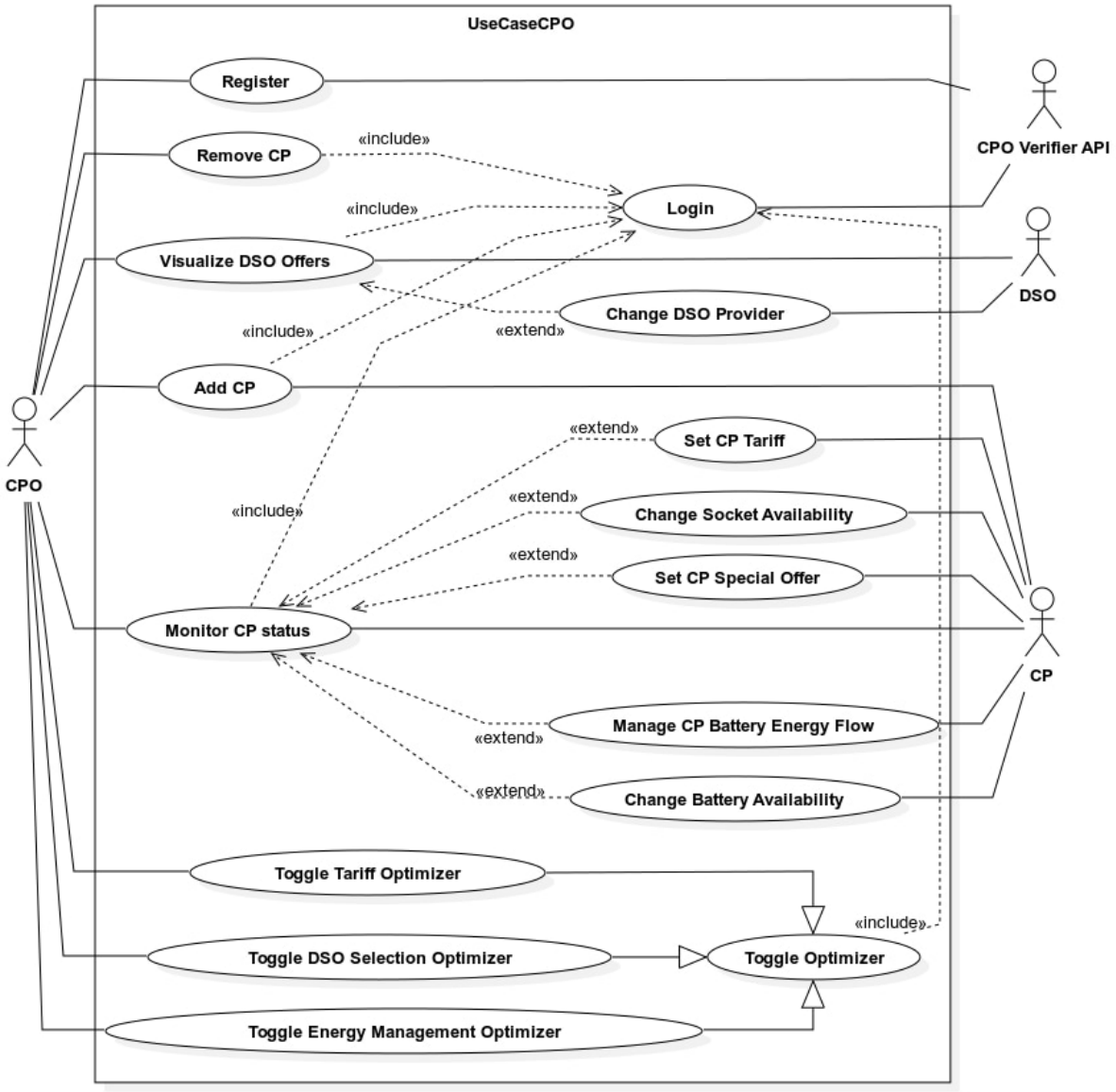

Figure 3.15: CPO Use Cases

ID	UC14
Name	CPO Registration
Actor	CPO, CPO Verifier API
Entry Condition	The CPO does not have an account and is on the login view
Event Flow	1. The CPO presses the "Register account" button; 2. The page with the registration form shows up; 3. The user compiles the required fields(CPOCode, Password, Confirm password); 4. The CPMS sends the CPOCode to an external verifier API; 5. The CPO Verifier API sends the confirmation to the CPMS; 6. The CPMS displays a success message.
Exit Conditions	The CPO account is created
Exceptions	1. The CPO does not enter all mandatory data; 2. The CPO data aren't valid; 3. The CPO already exists. • In all cases the CPMS will notify the CPO.

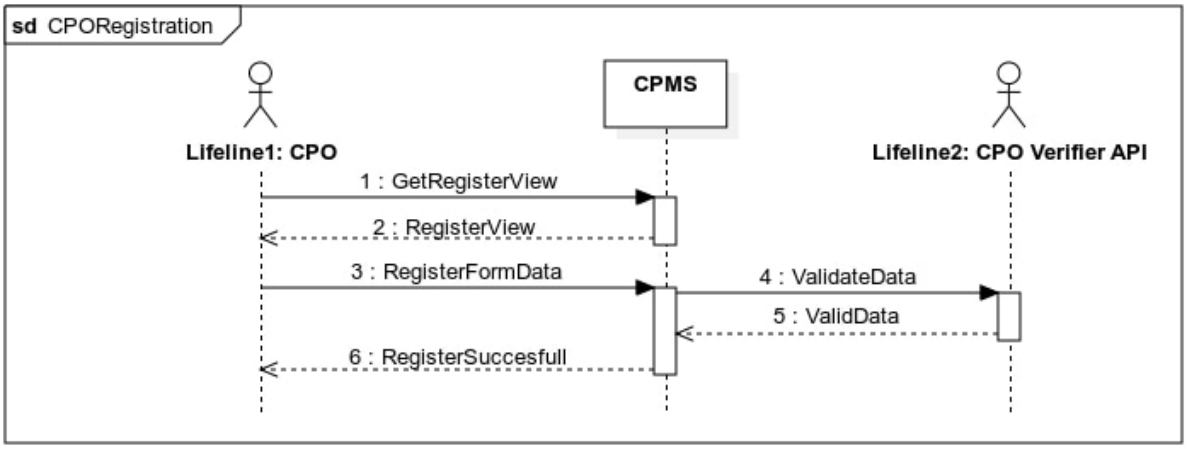

Figure 3.16: CPO Registration

ID	UC15
Name	**CPO Login**
Actor	CPO, CPO Verifier API
Entry Condition	The CPO is registered but not logged in and is on the login view
Event Flow	1. The user inserts the combination of his CPOCode and password; 2. The CPMS sends the data to the CPO Verifier API for verification; 3. The CPO Verifier API sends the confirmation to the CPMS; 4. The CPMS displays a success message.
Exit Conditions	The CPO is logged into the CPMS
Exceptions	1. The CPO does not enter his credentials before submitting; 2. The CPO enters invalid credentials. • In all cases the CPMS will notify the user.

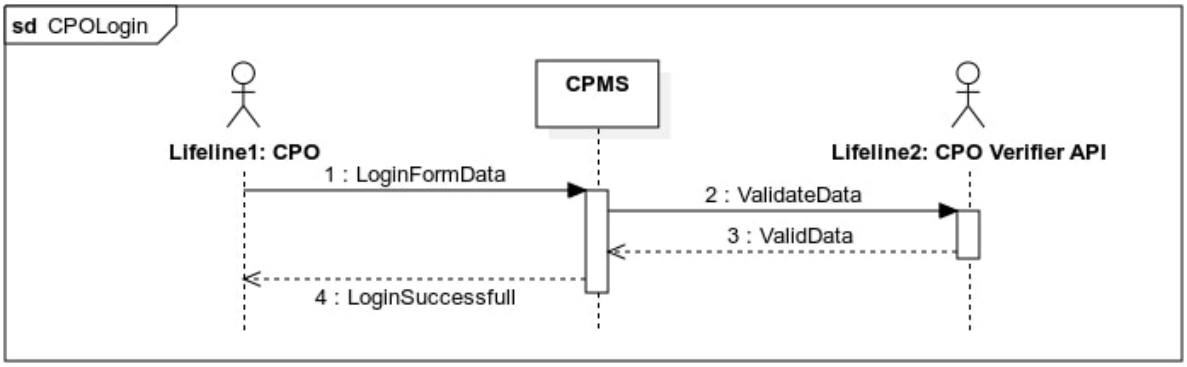

Figure 3.17: CPO Login

ID	UC16
Name	**Add CP**
Actor	CPO, CP
Entry Condition	The CPO is logged on the CPMS and is on the home page
Event Flow	1. The CPO presses the "Add charging point" button; 2. The CPMS displays a form to the CPO; 3. The CPO insert the location of the CP, a list of sockets together with their type, and the price of energy that will be applied in that CP; 4. The CPO inserts some information about how the CPMS will be able to connect to the new CP; 5. The CPMS establishes a connection to the new charging point; 6. The new charging point accepts the connection and acknowledges the CPMS; 7. The CPMS notify to all the eMSP of the presence of the new charging point.
Exit Conditions	The CPMS shows the home page to the CPO, where the new charging point is present
Exceptions	If the CPMS is not able to connect to the new charging point, it warns the CPO that can modify the connection parameters or remove the charging point

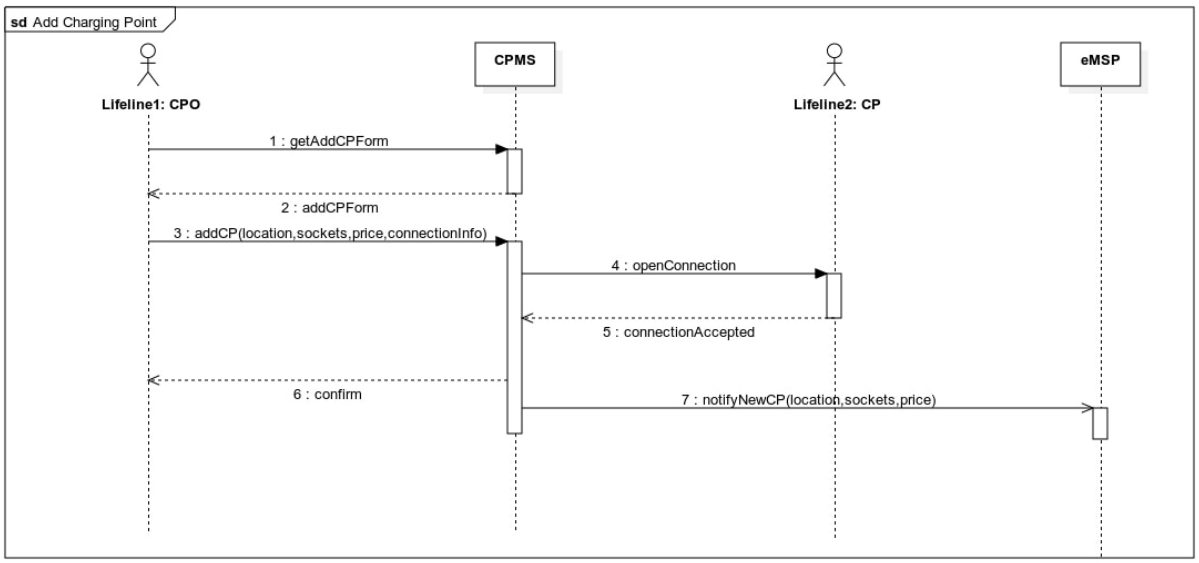

Figure 3.18: Add CP

ID	UC17
Name	**Manage CP's battery energy flow**
Actor	CPO, CP
Entry Condition	The CPO is logged on the CPMS, in the Manual Mode and is viewing the internal status of the charging point.
Event Flow	1. The CPO presses the "Manage Energy Source" button; 2. CPMS shows a page that contains details about the battery status and about the current energy sources; 3. The CPO presses the "Include battery" button; 4. The CPMS shows a form to the CPO; 5. The CPO inserts the percentage of energy that the charging point will take from the battery for the charging sessions and the minimum and maximum battery levels; 6. The CPMS notifies the charging point about the change of strategy for the energy flow; 7. The charging point confirms the update; 8. The CPMS shows a success message to the CPO.
Exit Conditions	The battery is added to the current energy sources.
Exceptions	If the minimum battery level is higher than the maximum battery level the CPMS will notify the CPO, which will have to change the parameters.

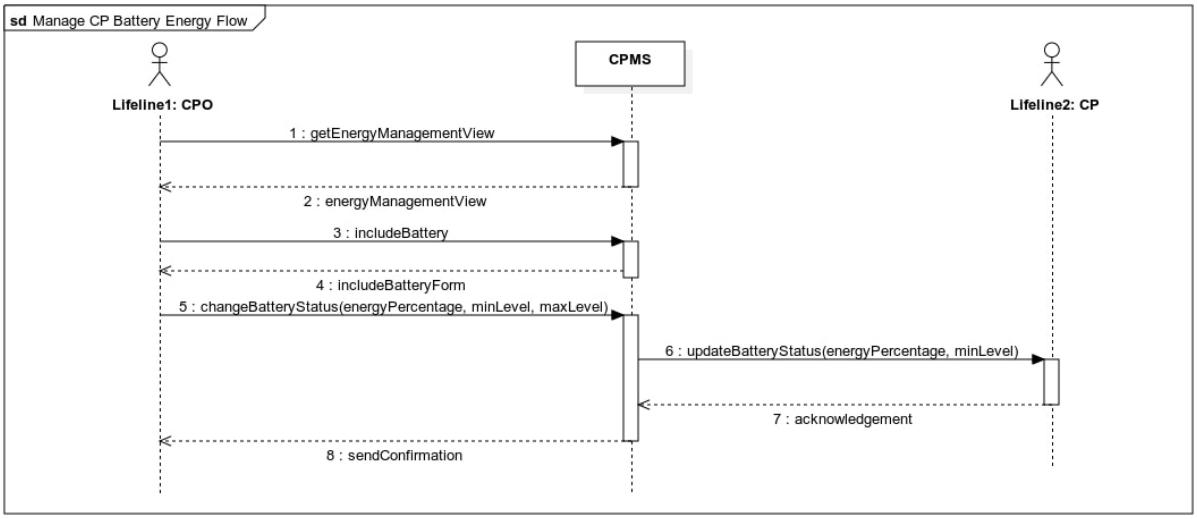

Figure 3.19: Manage CP's battery energy flow

ID	UC18
Name	**Change the socket's availability**
Actor	CPO, CP
Entry Condition	The CPO is logged on the CPMS and is on the home page
Event Flow	1. The CPO press the "CP View" button; 2. The CPMS shows a page containing CP information; 3. The CPO press the "Manage Sockets" button relative to a certain CP; 4. The CPMS shows a page containing all the sockets of a certain CP, including the type of sockets, the availability of the socket, how much energy the sockets are currently providing to the connected vehicles, and the estimated remaining time to complete the charging session; 5. The CPO clicks the "Change availability" button near a socket; 6. The CPMS shows a form where the CPO has to insert the starting date and time on which the socket will change the available; 7. The CPO inserts the date and the new availability; 8. The CPMS waits until the inserted time is reached; 9. The CPMS notifies the change of availability of a socket to the charging point; 10. The charging point confirms the update; 11. The CPMS sends a notification to the CPO to confirm the update; 12. The CPMS notifies all the connected eMSP about the update.
Exit Conditions	The CPMS returns to the page that shows the status of the sockets of the charging point
Exceptions	If there is a charging session that will terminate after the timestamp inserted by the CPO, the CPMS will send an error to the CPO

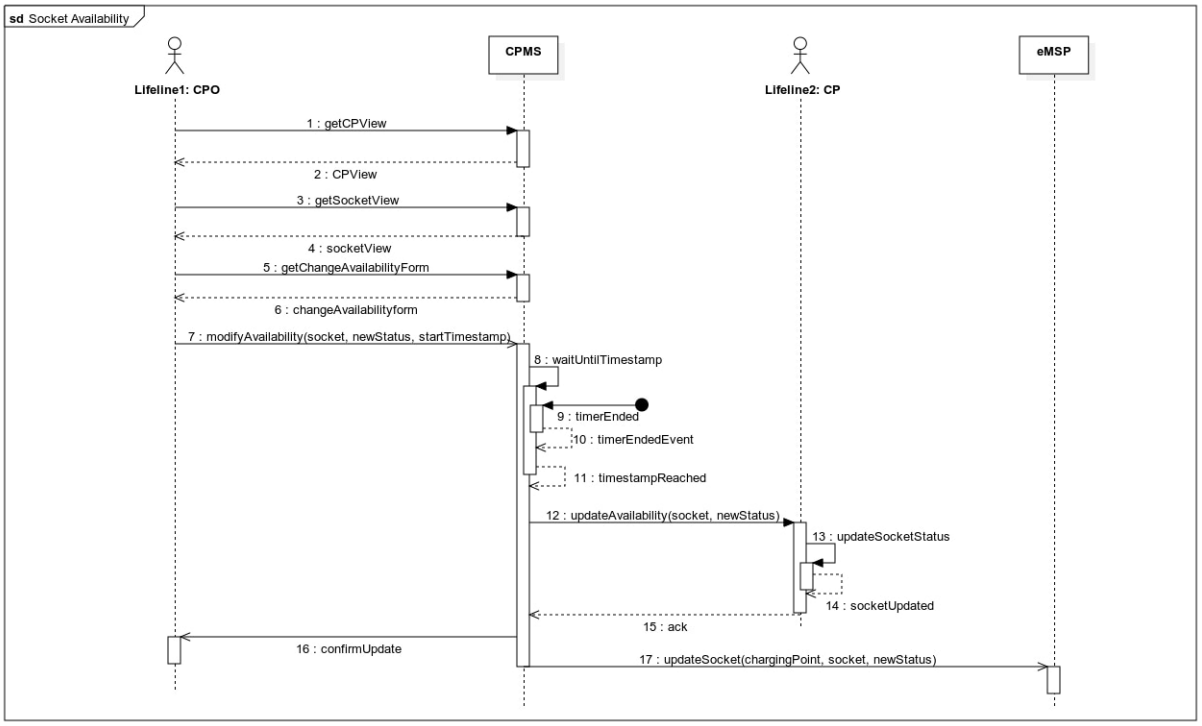

Figure 3.20: Change the socket's availability

ID	UC19
Name	Set charging point Tariff
Actor	CPO
Entry Condition	The CPO is logged on the CPMS and is on the home page
Event Flow	1. The CPO press the "CP View" button; 2. The CPMS shows a page containing CP information; 3. The CPO presses the "Price Update" button; 4. The CPMS shows a form to the CPO; 5. The CPO inserts the new price of the charging station and submits the changes; 6. The CPMS shows a success message; 7. The CPMS notifies all the connected eMSP about the change in the price.
Exit Conditions	The price of the charging point is changed
Exceptions	If the price that the CPO insert is lower than the current price of energy that the CPO is paying to the DSO, the CPMS warns the CPO before applying the change

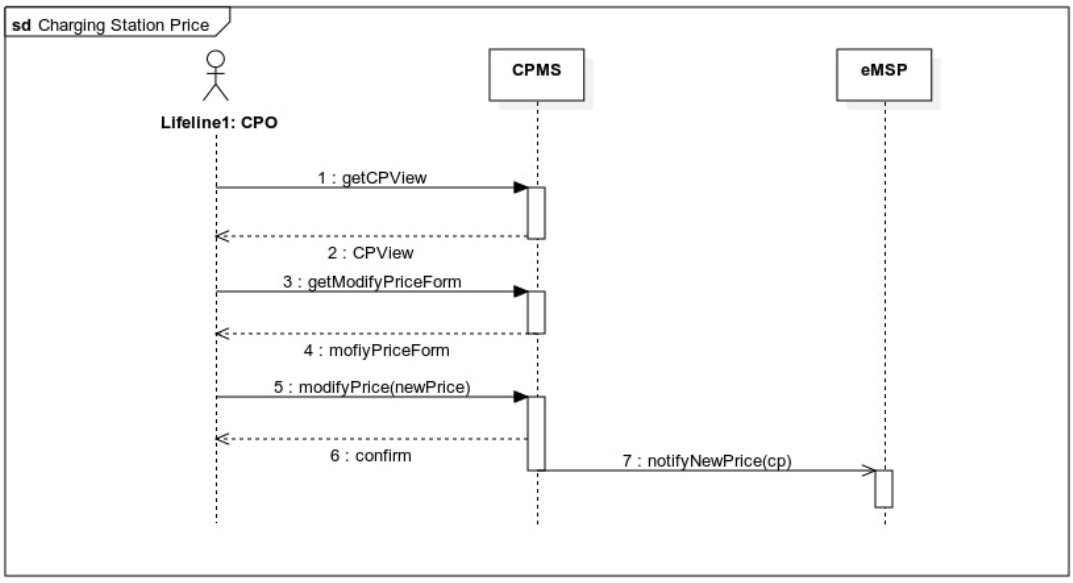

Figure 3.21: Set charging point Tariff

ID	UC20
Name	**Set charging point Special Offer**
Actor	CPO
Entry Condition	The CPO is logged on the CPMS and is viewing the home page
Event Flow	1. The CPO press the "CP View" button; 2. The CPMS shows a page containing CP information; 3. The CPO presses the "Add special offer" button; 4. The CPMS shows a form to the CPO; 5. The CPO inserts the new special offer for the charging station, which includes the price and the period on which the offer will be valid; 6. The CPMS shows a success message; 7. The CPMS notifies all the connected eMSP about the change in the price.
Exit Conditions	A new special offer is created
Exceptions	If the price that the CPO insert is lower than the current price of energy that the CPO is paying to the DSO, the CPMS warns the CPO before applying the change

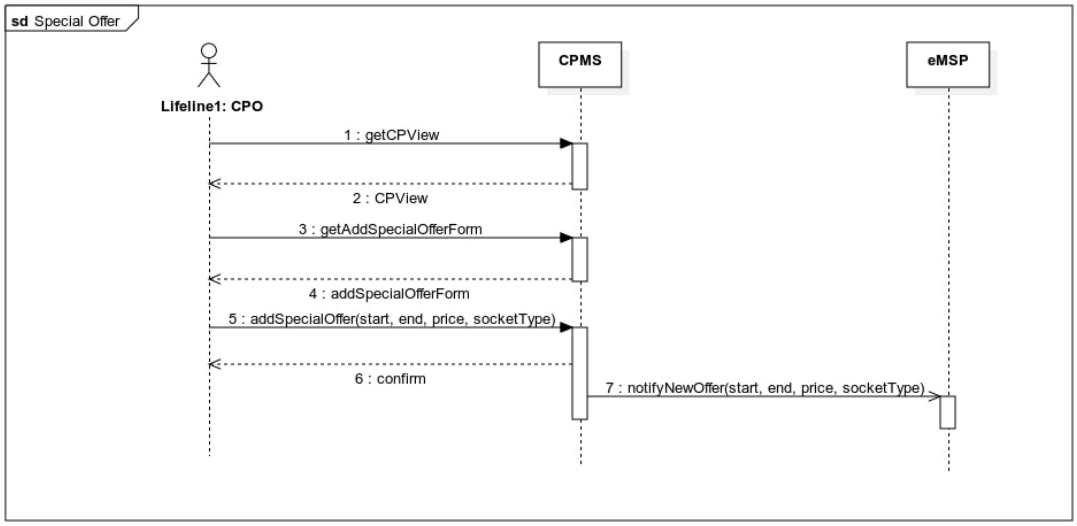

Figure 3.22: Set charging point Special Offer

ID	UC21
Name	Remove CP
Actor	CPO
Entry Condition	The CPO is logged in to the CPMS and is viewing the home page
Event Flow	1. The CPO presses the "Remove CP" button near a specific charging point; 2. The CPMS shows a confirmation popup to the CPO; 3. The CPO confirms the operation; 4. The CPMS closes the connection with the charging point; 5. The CPMS asynchronously notifies all the connected eMSP that the removed CP is no longer available.
Exit Conditions	The charging point is removed and the CPMS shows the home page to the CPO
Exceptions	If there are vehicles connected (or registered for a charging session that is not yet started) to some sockets of the charging point, the CPMS prevents the CPO from removing the charging point

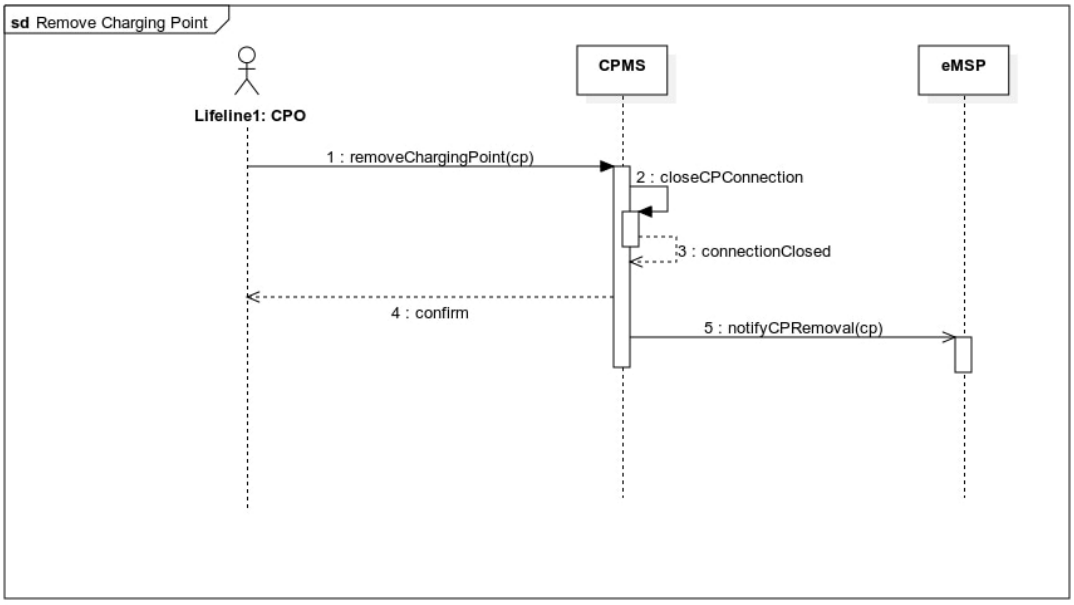

Figure 3.23: Remove CP

ID	UC22
Name	**Change DSO provider**
Actor	CPO, CP
Entry Condition	The CPO is logged in to the application and on and is on the main page view
Event Flow	1. The CPO selects a particular CP; 2. The CPMS displays general CP information; 3. The CPO press the "Manage Energy Source" button; 4. The CPMS displays all information in detail including energy source and battery information; 5. The CPO selects "DSOs' offers"; 6. The CPMS displays all available offers for that CP; 7. The CPO select the "Offer detail" option to know about the detail related to an offer; 8. The CPMS displays a page that includes all offers info and asks for a confirmation to change DSO provider; 9. The CPO reads the resume and confirms the option selected; 10. The CPMS sends a "change energy source" request to the CP; 11. The CP changes the energy source and confirms the changes to the CPMS; 12. The CPMS displays a confirmation message to the CPO.
Exit Conditions	The CPMS displays a success message to the CPO
Exceptions	1. The CP is unable to change the energy source; • An error message is displayed and the procedure is aborted. 2. The selected Energy offer is not able to provide the energy required to fulfill the CP's charging profile. • The CPMS displays a warning to the CP, asking for a change or confirmation.

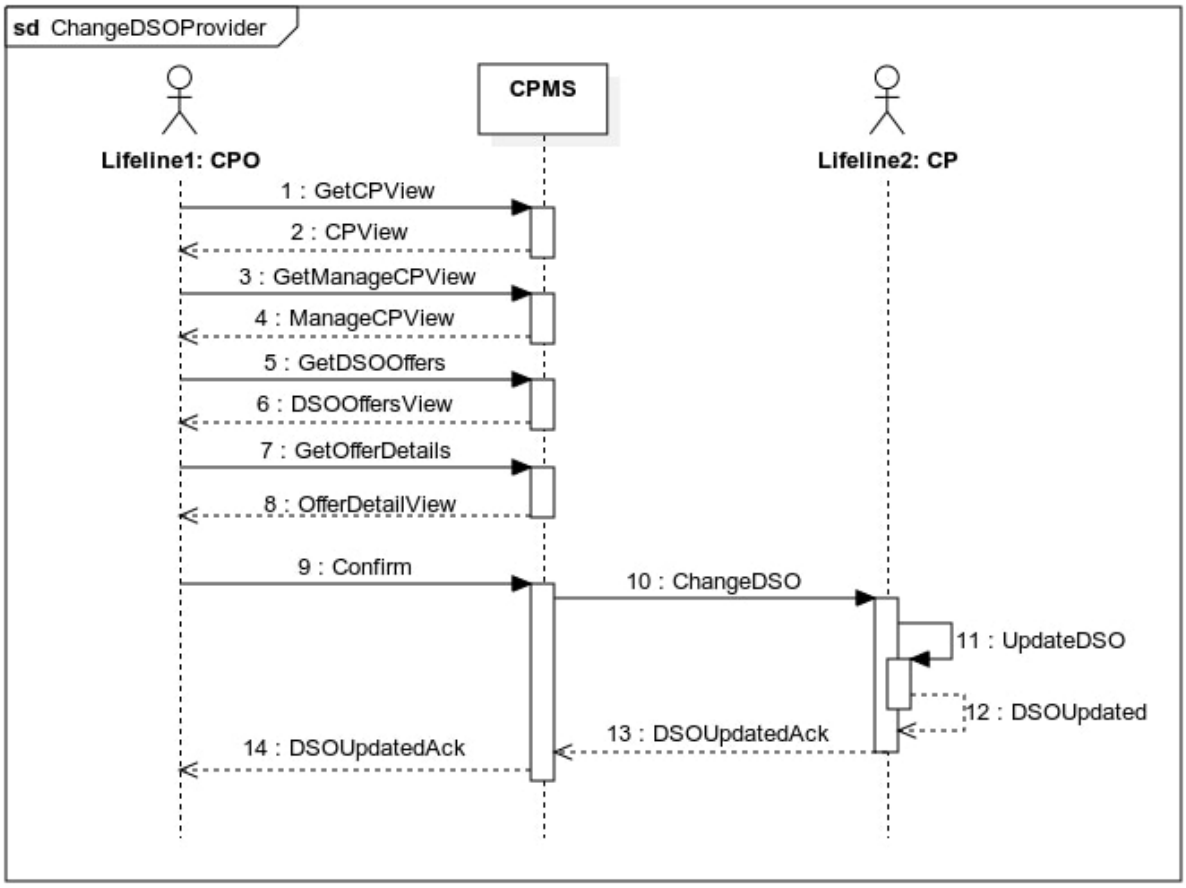

Figure 3.24: Change DSO provider

ID	UC23
Name	**Change Battery Availability**
Actor	CPO, CP
Entry Condition	The CPO is logged in to the application and on and is on the main page view
Event Flow	1. The CPO selects a particular CP; 2. The CPMS displays general CP information; 3. The CPO press the "Manage Energy Source" button; 4. The CPMS displays all information in detail including energy source and battery information; 5. The CPO switch presses the button "Exclude Battery" to make the battery not available; 6. The CPMS displays a message that asks the CPO to confirm the action; 7. The CP confirms; 8. The CPMS sends a "exclude battery request" to the CP; 9. The CP confirms the battery update and the system excludes the battery from the system; 10. The CPMS displays a success message.
Exit Conditions	The battery is excluded from the CP energy management system
Exceptions	1. There is no DSO offer for the selected CP; 2. The CP can't exclude battery.

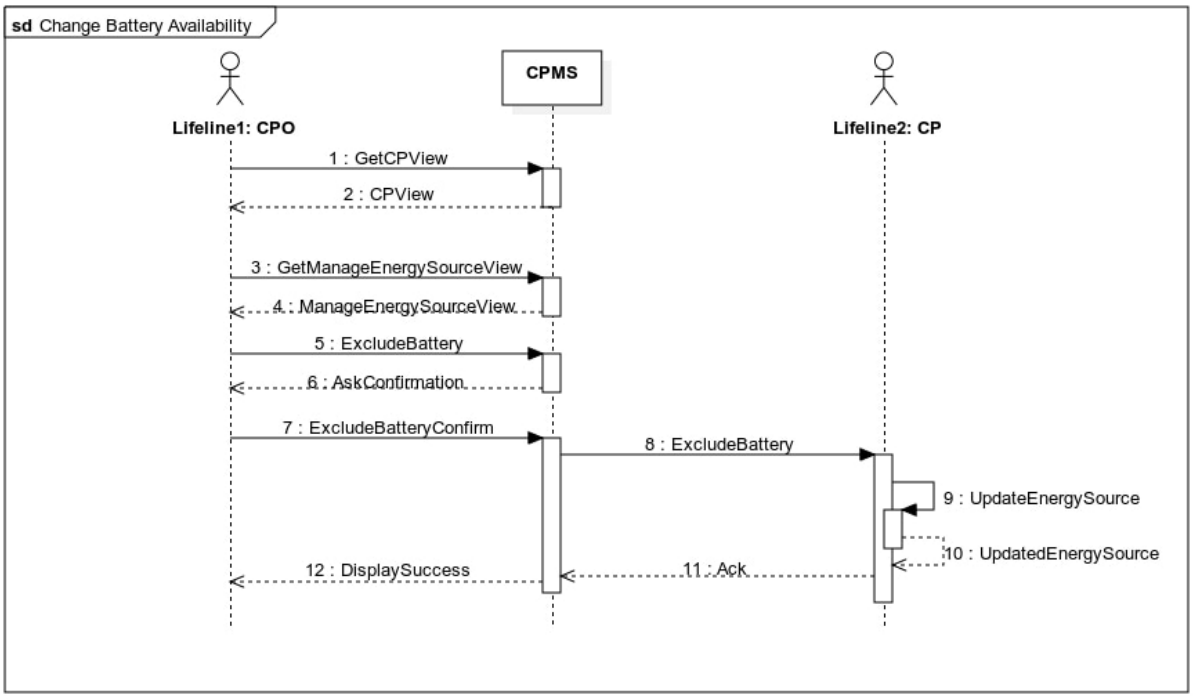

Figure 3.25: Change Battery Availability

ID	UC24
Name	**Toggle Tariff Optimizer**
Actor	CPO
Entry Condition	The CPO is logged in to the application and on and is on the main page view
Event Flow	1. The CPO selects a particular CP; 2. The CPMS displays general CP information; 3. The CPO press the "Optimizer" button; 4. The CPMS displays all the Optimizer settings; 5. The CPO compiles a form in order to select the optimizer settings for tariffs; 6. The CPMS displays a success message.
Exit Conditions	The optimizer switches its state to on if is off or off if is on
Exceptions	The CP doesn't exists.

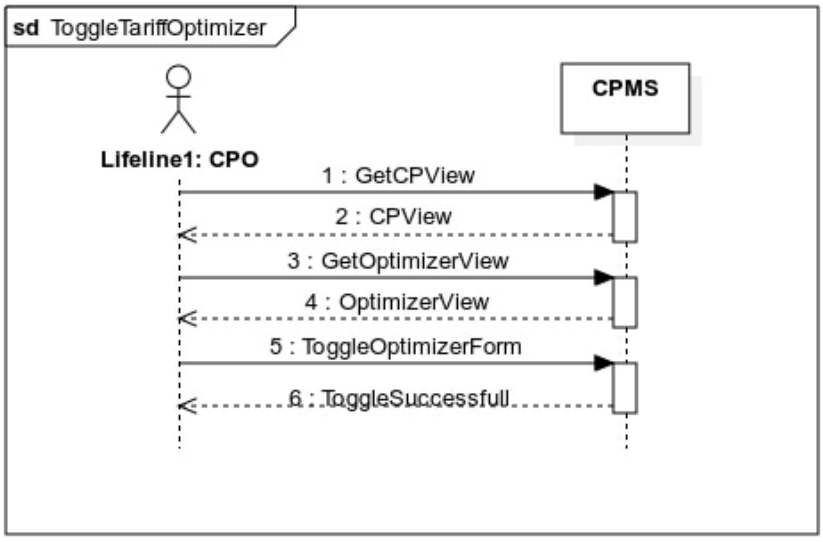

Figure 3.26: Toggle Tariff Optimizer

ID	UC25
Name	Toggle DSO Selection Optimizer
Actor	CPO
Entry Condition	The CPO is logged in to the application and on and is on the main page view
Event Flow	1. The CPO selects a particular CP; 2. The CPMS displays general CP information; 3. The CPO press the "Optimizer" button; 4. The CPMS displays all the Optimizer settings; 5. The CPO compiles a form in order to select the optimizer settings for DSO selection; 6. The CPMS displays a success message.
Exit Conditions	The optimizer switch its state to on if is off or off if is on
Exceptions	The CP doesn't exists.

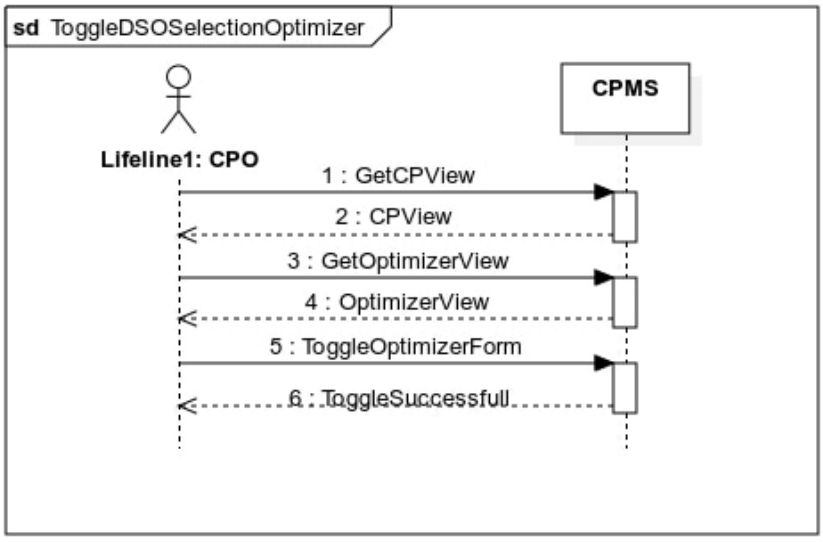

Figure 3.27: Toggle DSO Selection Optimizer

ID	UC26
Name	**Toggle Energy Management Optimizer**
Actor	CPO
Entry Condition	The CPO is logged in to the application and on and is on the main page view
Event Flow	1. The CPO selects a particular CP; 2. The CPMS displays general CP information; 3. The CPO press the "Optimizer" button; 4. The CPMS displays all the Optimizer settings; 5. The CPO compiles a form in order to select the optimizer settings for energy management; 6. The CPMS displays a success message.
Exit Conditions	The optimizer switch its state to on if is off or off if is on
Exceptions	The CP doesn't exists.

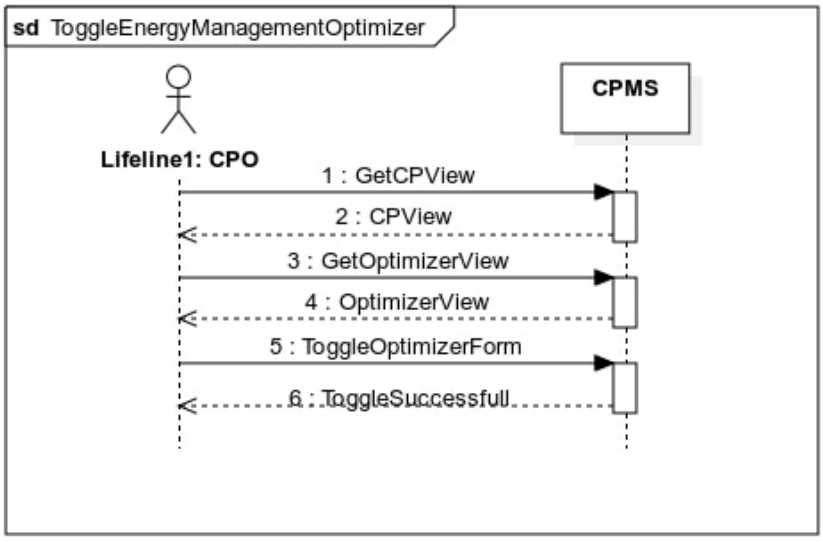

Figure 3.28: Toggle Energy Management Optimizer

3.2.3. Requirements

eMSP Requirements

R1	The eMSP shall allow unregistered users to register via e-mail and password.
R2	The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
R3	The eMSP shall allow the user to log in to the application using e-mail and password.
R4	The eMSP shall allow a user to register a new vehicle using the VIN code.
R5	The eMSP shall verify the validity of a VIN code inserted by a user.
R6	The eMSP shall allow the selection of a vehicle as the favourite of a user.
R7	The eMSP shall allow a user to remove a vehicle from the system and the relative data.
R8	The eMSP shall allow the User to add a payment method to the system.
R9	The eMSP shall verify the validity of the payment method inserted by the user.
R10	The eMSP shall allow the user to delete the payment method inserted.
R11	The eMSP shall prevent the user to reserve a socket if it has an unsolved payment.
R12	The eMSP shall prevent making a reservation if a reservation is active yet.
R13	The eMSP shall display an error message when failing to make a reservation.
R14	The system shall allow the user to visualize the CPs on a map in different ways distinguishing among the following case: 1. The CP has no socket compatible with the favourite vehicle; 2. The CP has one or more sockets compatible with the favourite vehicle but they are all busy; 3. The CP has one or more sockets free and compatible with the favourite vehicle. • For each of the previous options the user must distinguish whether CP has a special offer or not.
R15	The eMSP shall allow the user to reserve a free socket.
R16	The eMSP shall use the characteristic of the favourite vehicle to filter the results when it searches for CP.
R17	The eMSP shall interact with the relative CPMS when a user request to reserve a socket.
R18	The eMSP shall notify the user when a socket is reserved successfully.

R19	The eMSP shall allow the user to visualize the remaining time before the reservation expires.
R20	The eMSP shall allow the user to decline a suggestion.
R21	The eMSP shall make a new suggestion when a reservation generated from a suggestion fails.
R22	The eMSP shall use the VIN API to obtain the favourite vehicle's battery information including the current energy level.
R23	The eMSP shall access the user's calendar application to collect information about the user's schedule, including the time and position of the events inserted.
R24	The eMSP shall access the user's navigation system to collect information about the user's navigation path if it's active.
R25	The eMSP shall access the vehicle's GPS to retrieve its position.
R26	The eMSP shall notify the user via pop-up notification when a new suggestion is made.
R27	The eMSP shall allow the user to accept a suggestion.
R28	The eMSP when a suggestion is accepted shall reserve the socket reported in the suggestion
R29	The eMSP shall allow the user to cancel a reservation.
R30	The eMSP shall interact with the CPMS to cancel a reservation.
R31	The eMSP shall allow the user to start a charging process to a socket reserved by it.
R32	The eMSP shall allow the user to View the Charging Process when a socket is reserved successfully including information about battery status, the current price to pay, and the remaining time until the battery is fully recharged.
R33	The eMSP shall interact with the CPMS to start a charging session.
R34	The eMSP shall allow the user to pay when a charging process is complete.
R35	The eMSP shall collect payment information when process successfully a payment, saving information about the amount paid, the payment method used, the amount of energy used, the price of energy, the CP information such as location and CPO provider, the type of socket used (flow, fast or rapid), the duration of the Charging Session.
R36	The eMSP shall collect payment information also when the payment process fails, saving the payment in the system as an "unsolved payment".
R37	The eMSP shall display an error message to the user when the payment fails.
R38	The eMSP shall allow the user to pay the unsolved payment.

R39	The eMSP shall allow the user to set one among his payment method to pay.
R40	The eMSP shall interact with the CPMS to stop a charging session.
R41	The eMSP shall notify the User when the battery level of his vehicle is 100%.
R42	The eMSP shall allow the user to stop a charging process before the battery is full.
R43	The eMSP shall allow CPMSs to register in order to acquire relative information.
R44	The eMSP shall allow the user to visualize the charging process data when it's underway, including: • remaining time to complete the charge; • current battery level; • current energy power supplying; • current amount to pay.

CPMS Requirements

R45	The CPMS shall keep a socket reserved for twenty minutes after a User reserve it.
R46	The CPMS shall free the reserved socket if a charging process doesn't start within 20 minutes since it's reserved.
R47	The CPMS shall interact with the CP to reserve a Socket.
R48	The CPMS shall inform all eMSP(s) when a socket changes its status.
R49	The CPMS shall interact with the CP to cancel a reservation.
R50	The CPMS shall interact with the CP to start a charging session.
R51	The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
R52	The CPMS shall verify the CPO code via the CPO-verify API.
R53	The CPMS shall allow the CPO to log in with the CPO code.
R54	The CPMS shall verify the CPO code is valid when logging into the system.
R55	The CPMS shall allow the CPO to add a CP in the system.
R56	The CPMS shall inform all eMSP(s) when a CP is added.
R57	The CPMS shall interact with the CP to successfully complete its registration in the system.

R58	The CPMS shall allow the CPO to visualize information about his CPs, including: • Number of sockets available; • Current DSO used to acquire energy; • Batteries level; • Batteries status.
R59	The CPMS shall allow the CPO to change the availability of a socket.
R60	The CPMS shall interact with the CP to change the availability of a socket.
R61	The CPMS shall notify all eMSP(s) when a CP Tariff is updated.
R62	The CPMS shall allow the CPO to set the CP Energy Tariff.
R63	The CPMS shall allow the CPOs to create a special offer related to one or more CPs.
R64	The CPMS shall notify all eMSP(s) when a new offer is made.
R65	The CPMS shall allow the CPO to delete a CP from the system.
R66	The CPMS shall inform all eMSP(s) when a CP is removed.
R67	The CPMS shall allow the CPO to visualize DSOs' energy offers.
R68	The CPMS shall interact with the DSO to retrieve the Energy offers.
R69	The CPMS shall interact with the CP to change DSO energy source.
R70	The CPMS shall allow the CPO to change DSO.
R71	The CPMS shall allow the CPO to set one of the following configurations for the DSO selection policy for every CP: • **Automatic mode:** The system chooses from which DSOs acquire energy satisfying all charging profile (or as many as possible if not possible) requirements and minimizing cost. In this case, the CPMS prevents the user to change manually the energy acquisition configuration. • **Manual Mode:** The system allows the user to select manually from which DSO acquires energy.

R72	The CPMS shall allow the user to set one of the following configurations for the energy management for every CP: • **Automatic mode:** The system chooses the optimal mix between battery and DSO as the source of Energy. In this case, the CPMS shall prevent the user to change manually the energy management configuration. • **Manual Mode:** The system allow the user to manually manage the energy management configuration.
R73	The CPMS shall interact with the CP to stop a charging session.
R74	The CPMS shall allow the CPO to change the availability of a battery of a CP, excluding/including it from the CP's possible "Current Energy Source".
R75	The CPMS shall not consider unavailable batteries as a possible source of energy when the Energy Management optimizer is set in Automatic Mode.
R76	The CPMS shall prevent the CPO from setting an unavailable battery as one of the current sources of energy when the Energy Management optimizer is set in Manual Mode.
R77	The CPMS shall assign for each socket a maximum energy request based on the socket types.
R78	The CPMS shall interact with the CP to change battery availability.
R79	The CPMS shall allow the CPO to set the following parameters for the battery (or batteries if more than on included) of CPs that have the configuration of the energy flow management set to "Manual": Minimum Energy trigger, Maximum Energy trigger, Energy Mix.
R80	The CPMS shall interact with the CP to change the battery flow configuration.
R81	The CPMS shall register to eMSP(s) to exchange information.
R82	The CPMS shall recompute a charging profile for each socket of a CP every time the DSO relative to the CP change, to guarantee that all socket's minimum energy requests can be satisfied at the same time.
R83	The CPMS shall allow the CPO to set the Tariff policy of a certain CP in one of the following: 1. **Automatic:** The Tariff is chosen by the CPMS in order to choose competitive and profitable prices. 2. **Manual:** The Tariff is set by the CPO.

3.2.4. Goal mapping on requirements

G1 Allow users to visualize nearby charging stations and the relative price of energy, socket availability, and if they provide offers;

- **R1.** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R14.** The system shall allow the user to visualize the CPs on a map in different ways distinguishing among the following case:
  (a) The CP has no socket compatible with the favourite vehicle;
  (b) The CP has one or more sockets compatible with the favourite vehicle but they are all busy;
  (c) The CP has one or more sockets free and compatible with the favourite vehicle.
  – For each of the previous options the user must distinguish whether CP has a special offer or not.
- **R16.** The eMSP shall use the characteristic of the favourite vehicle to filter the results when it searches for CP.
- **R25.** The eMSP shall access the user's GPS to retrieve his position.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.
- **R48.** The CPMS shall inform all eMSP(s) when a socket changes its status.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R61.** The CPMS shall notify all eMSP(s) when a CP Tariff is updated.
- **R64.** The CPMS shall notify all eMSP(s) when a new offer is made.

- **R66.** The CPMS shall inform all eMSP(s) when a CP is removed.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D7.** All the sockets of a certain type in the same charging point have the same tariff.

G2 Allow users to reserve a charge at a certain charging point for a certain time slot;

- **R1** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R11.** The eMSP shall prevent the user to reserve a socket if it has an unsolved payment.
- **R12.** The eMSP shall prevent making a reservation if a reservation is active yet.
- **R13.** The eMSP shall display an error message when failing to make a reservation.
- **R15.** The eMSP shall allow the user to reserve a free socket.
- **R17.** The eMSP shall interact with the relative CPMS when a user requests to reserve a socket.
- **R18.** The eMSP shall notify the user when a socket is reserved successfully.
- **R19.** The eMSP shall allow the user to visualize the remaining time before the reservation expires.
- **R28.** The eMSP when a suggestion is accepted shall reserve the socket reported in the suggestion.
- **R29.** The eMSP shall allow the user to cancel a reservation.
- **R30.** The eMSP shall interact with the CPMS to cancel a reservation.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.

- **R45.** The CPMS shall keep a socket reserved for twenty minutes after a User reserve it.
- **R46.** The CPMS shall free the reserved socket if a charging process doesn't start within 20 minutes since it's reserved.
- **R47.** The CPMS shall interact with the CP to reserve a Socket.
- **R48.** The CPMS shall inform all eMSP(s) when a socket changes its status.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D2.** The user can always physically access the socket in the reserved time slot.
- **D12.** A reservation is valid from the moment on which is made to the following 20 minutes, after which it is cancelled.

G3 Allow the user to start a charging process at a station;

- **R1** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R31.** The eMSP shall allow the user to start a charging process to a socket reserved by it.
- **R33.** The eMSP shall interact with the CPMS to start a charging session.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.
- **R50.** The CPMS shall interact with the CP to start a charging session.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D2.** The user can always physically access the socket in the reserved time slot.
- **D10.** The CP is able to detect when a user plugs/unplugs the socket.
- **D13.** A charging session can start only from an existing valid reservation.

G4 Allow the user to monitor the status of the charging process with valuable data.

- **R1** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R32.** The eMSP shall allow the user to View the Charging Process when a socket is reserved successfully including information about battery status, the current price to pay, and the remaining time until the battery is fully recharged.
- **R40.** The eMSP shall interact with the CPMS to stop a charging session.
- **R41.** The eMSP shall notify the User when the battery level of his vehicle is 100%.
- **R42.** The eMSP shall allow the user to stop a charging process before the battery is full.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.
- **R44.** The eMSP shall allow the user to visualize the charging process data when it's underway, including:
  - remaining time to complete the charge;
  - current battery level;
  - current energy power supplying;
  - current amount to pay.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D1.** The sockets have their own sensor that is able to estimate the status of the battery of the vehicle that is connected.
- **D15.** The CP automatically stops the energy flow when it detects that the battery of the car is full.

G5 Allow the user to pay for the service;

- **R1** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R8.** The eMSP shall allow the User to add a payment method to the system.
- **R9.** The eMSP shall verify the validity of the payment method inserted by the user.
- **R10.** The eMSP shall allow the user to delete the payment method inserted.
- **R34.** The eMSP shall allow the user to pay when a charging process is complete.
- **R35.** The eMSP shall collect payment information when process successfully a payment, saving information about the amount paid, the payment method used, the amount of energy used, the price of energy, the CP information such as location and CPO provider, the type of socket used (flow, fast or rapid), the duration of the Charging Session.
- **R36.** The eMSP shall collect payment information also when the payment process fails, saving the payment in the system as an "unsolved payment".
- **R37.** The eMSP shall display an error message to the user when the payment fails.
- **R38.** The eMSP shall allow the user to pay the unsolved payment.
- **R39.** The eMSP shall allow the user to set one among his payment method to pay.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D4.** There exists an external service that checks the validity of billing information.

- **D7.** All the sockets of a certain type in the same charging point have the same tariff.

G6 Gives users suggestions about the optimal schedule to charge the vehicle depending on battery status, the schedule of the users, and offers provided by CPOs.

- **R1** The eMSP shall allow unregistered users to register via e-mail and password.
- **R2.** The eMSP shall send a confirmation mail when a new user tries to register to it containing a code to confirm the account.
- **R3.** The eMSP shall allow the user to log in to the application using e-mail and password.
- **R4.** The eMSP shall allow a user to register a new vehicle using the VIN code.
- **R5.** The eMSP shall verify the validity of a VIN code inserted by a user.
- **R6.** The eMSP shall allow the selection of a vehicle as the favourite of a user.
- **R7.** The eMSP shall allow a user to remove a vehicle from the system and the relative data.
- **R20.** The eMSP shall allow the user to decline a suggestion.
- **R21.** The eMSP shall make a new suggestion when a reservation generated from a suggestion fails.
- **R22.** The eMSP shall use the VIN API to obtain the favourite vehicle's battery information including the current energy level.
- **R23.** The eMSP shall access the user's calendar application to collect information about the user's schedule, including the time and position of the events inserted.
- **R24.** The eMSP shall access the user's navigation system to collect information about the user's navigation path if it's active.
- **R25.** The eMSP shall access the vehicle's GPS to retrieve its position.
- **R26.** The eMSP shall notify the user via pop-up notification when a new suggestion is made.
- **R27.** The eMSP shall allow the user to accept a suggestion.

- **R28.** The eMSP when a suggestion is accepted shall reserve the socket reported in the suggestion.
- **R43.** The eMSP shall allow CPMSs to register in order to acquire relative information.
- **R56.** The CPMS shall inform all eMSP(s) when a CP is added.
- **R81.** The CPMS shall register to eMSP(s) to exchange information.
- **D5.** There exists an external service that checks the validity of the VIN code of the vehicles.
- **D6.** There exists an external service that retrieves the status of the battery of a vehicle when a socket is not plugged.
- **D12.** A reservation is valid from the moment on which is made to the following 20 minutes, after which it is cancelled.

G7 Allow CPOs dynamically decide from which DSO to acquire energy providing information on energy prices.

- **R51.** The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
- **R52.** The CPMS shall verify the CPO code via CPO-verify API.
- **R53.** The CPMS shall allow the CPO to log in with the CPO code.
- **R54.** The CPMS shall verify the CPO code is valid when logging into the system.
- **R55.** The CPMS shall allow the CPO to add a CP to the system.
- **R57.** The CPMS shall interact with the CP to successfully complete its registration in the system.
- **R58.** The CPMS shall allow the CPO to visualize information about his CPs, including:
  - Number of sockets available;
  - Current DSO used to acquire energy;
  - Batteries level;
  - Batteries status.

- **R67.** The CPMS shall allow the CPO to visualize DSOs' energy offers.
- **R68.** The CPMS shall interact with the DSO to retrieve the Energy offers.
- **R69.** The CPMS shall interact with the CP to change DSO energy source.
- **R70.** The CPMS shall allow the CPO to change DSO.
- **R71.** The CPMS shall allow the CPO to set one of the following configurations for the DS selection policy for every CP:
  - Automatic mode: The system chooses from which DSOs acquire energy satisfying all charging profile (or as many as possible if not possible) requirements and minimizing cost. In this case, the CPMS prevents the user to change manually the energy acquisition configuration.
  - Manual Mode: The system allows the user to select manually from which DSO acquires energy.
- **D3.** There exists a sensor of the CP that is able to monitor the quantity of energy that the DSO is currently providing.
- **D8.** A CPO is identified by a unique identifier provided by the local authorities.
- **D9.** A CP is identified by a unique identifier provided by the local authorities.
- **D11.** Each CP can have at most one DSO that provides the energy at a time.
- **D14.** For each charging point there exists at least one DSO that can provide the energy to it.

G8 Allow CPOs dynamically decide the cost of charging and when setting special offers.

- **R51.** The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
- **R52.** The CPMS shall verify the CPO code via CPO-verify API.
- **R53.** The CPMS shall allow the CPO to log in with the CPO code.
- **R54.** The CPMS shall verify the CPO code is valid when logging into the system.
- **R55.** The CPMS shall allow the CPO to add a CP to the system.
- **R57.** The CPMS shall interact with the CP to successfully complete its registration in the system.

- **R58.** The CPMS shall allow the CPO to visualize information about his CPs, including:
  - Number of sockets available;
  - Current DSO used to acquire energy;
  - Batteries level;
  - Batteries status.
- **R62.** The CPMS shall allow the CPO to set the CP Energy Tariff.
- **R63.** The CPMS shall allow the CPOs to create a special offer related to one or more CPs.
- **R83.** The CPMS shall allow the CPO to set the Tariff policy of a certain CP in one of the following:
  - **Automatic:** The Tariff is chosen by the CPMS in order to choose competitive and profitable prices.
  - **Manual:** The Tariff is set by the CPO.
- **D7.** All the sockets of a certain type in the same charging point have the same tariff.
- **D14.** For each charging point there exists at least one DSO that can provide the energy to it.

G9 Allow CPOs dynamically decide whether or not to store energy in their internal batteries.

- **R51.** The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
- **R52.** The CPMS shall verify the CPO code via CPO-verify API.
- **R53.** The CPMS shall allow the CPO to log in with the CPO code.
- **R54.** The CPMS shall verify the CPO code is valid when logging into the system.
- **R55.** The CPMS shall allow the CPO to add a CP to the system.
- **R57.** The CPMS shall interact with the CP to successfully complete its registration in the system.

- **R58.** The CPMS shall allow the CPO to visualize information about his CPs, including:
  - Number of sockets available;
  - Current DSO used to acquire energy;
  - Batteries level;
  - Batteries status.
- **R72.** The CPMS shall allow the user to set one of the following configurations for the energy management for every CP:
  - Automatic mode: The system chooses the optimal mix between battery and DSO as the source of Energy. In this case, the CPMS shall prevent the user to change manually the energy management configuration.
  - Manual Mode: The system allow the user to manually manage the energy management configuration.
- **R74.** The CPMS shall allow the CPO to change the availability of a battery of a CP, excluding/including it from the CP's possible "Current Energy Source".
- **R75.** The CPMS shall not consider unavailable batteries as a possible source of energy when the Energy Management optimizer is set in Automatic Mode.
- **R76.** The CPMS shall prevent the CPO from setting an unavailable battery as one of the current sources of energy when the Energy Management optimizer is set in Manual Mode.
- **R78.** The CPMS shall interact with the CP to change battery availability.
- **R79.** The CPMS shall allow the CPO to set the following parameters for the battery (or batteries if more than on included) of CPs that have the configuration of the energy flow management set to "Manual": Minimum Energy trigger, Maximum Energy trigger, Energy Mix.
- **R80.** The CPMS shall interact with the CP to change the battery flow configuration.
- **D3.** There exists a sensor of the CP that is able to monitor the quantity of energy that the DSO is currently providing.

G10 Allow CPOs dynamically decide, during a charging process, to use the stored energy in the battery or to acquire directly from DSOs to fulfil the charge or a mix of both.

- **R51.** The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
- **R52.** The CPMS shall verify the CPO code via CPO-verify API.
- **R53.** The CPMS shall allow the CPO to log in with the CPO code.
- **R54.** The CPMS shall verify the CPO code is valid when logging into the system.
- **R55.** The CPMS shall allow the CPO to add a CP to the system.
- **R57.** The CPMS shall interact with the CP to successfully complete its registration in the system.
- **R58.** The CPMS shall allow the CPO to visualize information about his CPs, including:
  - Number of sockets available;
  - Current DSO used to acquire energy;
  - Batteries level;
  - Batteries status.
- **R67.** The CPMS shall allow the CPO to visualize DSOs' energy offers.
- **R68.** The CPMS shall interact with the DSO to retrieve the Energy offers.
- **R69.** The CPMS shall interact with the CP to change DSO energy source.
- **R70.** The CPMS shall allow the CPO to change DSO.
- **R71.** The CPMS shall allow the CPO to set one of the following configurations for the DS selection policy for every CP:
  - Automatic mode: The system chooses from which DSOs acquire energy satisfying all charging profile (or as many as possible if not possible) requirements and minimizing cost. In this case, the CPMS prevents the user to change manually the energy acquisition configuration.
  - Manual Mode: The system allows the user to select manually from which DSO acquires energy.
- **R72.** The CPMS shall allow the user to set one of the following configurations for the energy management for every CP:

  - Automatic mode: The system chooses the optimal mix between battery and DSO as the source of Energy. In this case, the CPMS shall prevent the user to change manually the energy management configuration.
  - Manual Mode: The system allow the user to manually manage the energy management configuration.

- **R74.** The CPMS shall allow the CPO to change the availability of a battery of a CP, excluding/including it from the CP's possible "Current Energy Source".
- **R75.** The CPMS shall not consider unavailable batteries as a possible source of energy when the Energy Management optimizer is set in Automatic Mode.
- **R76.** The CPMS shall prevent the CPO from setting an unavailable battery as one of the current sources of energy when the Energy Management optimizer is set in Manual Mode.
- **R78.** The CPMS shall interact with the CP to change battery availability.
- **R79.** The CPMS shall allow the CPO to set the following parameters for the battery (or batteries if more than on included) of CPs that have the configuration of the energy flow management set to "Manual": Minimum Energy trigger, Maximum Energy trigger, Energy Mix.
- **R80.** The CPMS shall interact with the CP to change the battery flow configuration.
- **D3.** There exists a sensor of the CP that is able to monitor the quantity of energy that the DSO is currently providing.
- **D11.** Each CP can have at most one DSO that provides the energy at a time.
- **D14.** For each charging point there exists at least one DSO that can provide the energy to it.

G11 Allow the CPOs to Manage their CPs and the relative socket in order to enable drivers to use them.

- **R17.** The eMSP shall interact with the relative CPMS when a user request to reserve a socket.
- **R30.** The eMSP shall interact with the CPMS to cancel a reservation.
- **R33.** The eMSP shall interact with the CPMS to start a charging session.
- **R40.** The eMSP shall interact with the CPMS to stop a charging session.

- **R45.** The CPMS shall keep a socket reserved for twenty minutes after a User reserve it.
- **R47.** The CPMS shall interact with the CP to reserve a Socket.
- **R49.** The CPMS shall interact with the CP to cancel a reservation.
- **R50.** The CPMS shall interact with the CP to start a charging session.
- **R51.** The CPMS shall allow the CPO to register to the system via CPO-code and CPO-verify API.
- **R52.** The CPMS shall verify the CPO code via CPO-verify API.
- **R53.** The CPMS shall allow the CPO to log in with the CPO code.
- **R54.** The CPMS shall verify the CPO code is valid when logging into the system.
- **R55.** The CPMS shall allow the CPO to add a CP to the system.
- **R57.** The CPMS shall interact with the CP to successfully complete its registration in the system.
- **R58.** The CPMS shall allow the CPO to visualize information about his CPs, including:
  - Number of sockets available;
  - Current DSO used to acquire energy;
  - Batteries level;
  - Batteries status.
- **R59.** The CPMS shall allow the CPO to change the availability of a socket.
- **R60.** The CPMS shall interact with the CP to change the availability of a socket.
- **R65.** The CPMS shall allow the CPO to delete a CP from the system.
- **R73.** The CPMS shall interact with the CP to stop a charging session.
- **R77.** The CPMS shall assign for each socket a maximum energy request based on the socket types.
- **R82.** The CPMS shall recompute a charging profile for each socket of a CP every time the DSO relative to the CP change, to guarantee that all socket's

minimum energy requests can be satisfied at the same time.

- **D1.** The sockets have their own sensor that is able to estimate the status of the battery of the vehicle that is connected.
- **D7.** All the sockets of a certain type in the same charging point have the same tariff.
- **D10.** The CP is able to detect when a user plugs/unplugs the socket.
- **D12.** A reservation is valid from the moment on which is made to the following 20 minutes, after which it is canceled.
- **D15.** The CP automatically stops the energy flow when it detects that the battery of the car is full.

3.2.5. Traceability matrix

Use Cases Mapping

UC1	R1, R2
UC2	R3
UC3	R4, R5
UC4	R6
UC5	R7
UC6	R8, R9
UC7	R10
UC8	R11, R12, R13, R14, R15, R16, R17, R18, R19, R43, R45, R47, R48, R81
UC9	R11, R12, R13, R15, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R43, R45, R47, R48, R81
UC10	R29, R30, R43, R46, R48, R49, R81
UC11	R31, R32, R33, R43, R44, R50, R81
UC12	R40, R41, R42, R43, R44, R73, R81
UC13	R34, R35, R36, R37, R38, R39
UC14	R51, R52
UC15	R53, R54
UC16	R43, R55, R56, R57, R81
UC17	R58, R79, R80
UC18	R43, R48, R58, R59, R60, R81
UC19	R43, R58, R61, R62, R81
UC20	R43, R58, R63, R64, R81
UC21	R43, R65, R66, R81
UC22	R58, R67, R68, R69, R70, R82
UC23	R58, R74, R75, R76, R78
UC24	R83
UC25	R71
UC26	R72

Goals Mapping

Goals	Requirements	DomainAssumptions
G1	R1, R2, R3, R14, R16, R25, R43, R48, R56, R61, R64, R66, R81	D7
G2	R1, R2, R3, R11, R12, R13, R15, R17, R18, R19, R28, R29, R30, R43, R45, R46, R47, R48, R56, R81	D2, D12
G3	R1, R2, R3, R31, R33, R43, R50, R56, R81	D2, D13, D10
G4	R1, R2, R3, R32, R40, R41, R42, R43, R44, R56, R81	D1, D15
G5	R1, R2, R3, R8, R9, R10, R34, R35, R36, R37, R38, R39, R43, R56, R81	D4, D7
G6	R1, R2, R3, R4, R5, R6, R7, R20, R21, R22, R23, R24, R25, R26, R27, R28, R43, R56, R81	D5, D6, D12
G7	R51, R52, R53, R54, R55, R57, R58, R67, R68, R69, R70, R71	D3, D8, D9, D11, D14
G8	R51, R52, R53, R54, R55, R57, R58, R62, R63, R83	D7, D14
G9	R51, R52, R53, R54, R55, R57, R58, R72, R74, R75, R76, R78, R79, R80	D3
G10	R51, R52, R53, R54, R55, R57, R58, R67, R68, R69, R70, R71, R72, R74, R75, R76, R78, R79, R80	D3, D11, D14
G11	R17, R30, R33, R40, R45, R47, R49, R50, R51, R52, R53, R54, R55, R57, R58, R59, R60, R65, R73, R77, R82	D1, D7, D10, D12, D15

3.3. Performance Requirements

3.3.1. eMSP

The eMSP must be able to handle a high number of concurrent requests. In addition, the eMSP must handle also many concurrent updates of the status of the charging points from the CPMSs that are connected to it. In particular, the functionalities that are mostly affected by these concurrency issues:

- Stations researches, since many users can do them in moments that are really close to some status update of the charging points. The system should be designed in such a way as to reduce the response time and, at the same time, to provide the most recent status of the returned charging points.
- Reservations, since many users can try to make a reservation on the same socket at the same time.

Despite the expected high workload, the eMSP should have a good response time, typically less or equal to 2 seconds, in order to provide a better experience to the users and not make them think that the service is unavailable. All the computations must be done on the server side and the client applications shall be as lightweight as possible.
Another important performance aspect regards the suggestions: when the battery of the "Favourite Vehicle" of a user is at a low level, the eMSP should realize it as soon as possible in order to make effective suggestions. This implies that the battery status information must be collected frequently, especially when the user is driving his car.

3.3.2. CPMS

The CPMS is responsible to handle a high amount of updates for every socket of every CP regarding different aspects. Every single update has to be sent to every connected eMSP as soon as possible in order to guarantee that they have the latest data. The various updates sent from the CPMS to the eMSPs are regarding:

- Every change in the status socket as available, reserved, charging, unavailable, ...;
- Every change in the tariffs of each type of socket;
- Every new special offer that is active at the moment;
- Every full charge notification to the relative eMSP.

In order to achieve consistency of the data and reduce the traffic between the systems,

some of the updates only happen if an eMSP makes a request to the CPMS, requests like the actual status of the reservation or the charging process with their relative information. All the computations must be done on the server side so all the updates are handled as fast as possible.

3.4. Design Constraints

3.4.1. Standards Compliance

The eMSP must be compliant with the GDPR's law for managing the users' data and must provide secure communication with the user for sending and retrieving his data. The user must read and accept the privacy policy, otherwise, he won't be able to use the service.
The CPMS should follow the guidelines imposed by the standard protocol used to communicate with the external components on which it relies to provide its functions: the charging points (OCPP) and the DSOs (OSCP and OpenADR).
Both the eMSP and the CPMS must comply with the OCPI standard; this should allow the eMSP to be connected by many CPMSs and the CPMS to be connected to many eMSPs.

3.4.2. Hardware Limitations

In order to use the eMSP, each user is required to have a mobile device from which he can access the application. In addition, to receive suggestions the user must have a device that is able to detect his GPS position.
Regarding the CPOs, each of those must have a device with a modern web browser in order to be able to connect to the CPMS.
Both the users of the eMSP and the CPOs must use a device with an internet connection.

3.5. Software System Attributes

3.5.1. Reliability

Both the eMSP and the CPMS must be highly reliable and must prevent downtime. With this in mind, the two systems must be fault tolerant in order to prevent data inconsistencies.

3.5.2. Availability

The eMSP must be highly available in order to make the users have a better experience, the minimum value for it should be 99,5%.
The CPMS shall guarantee a 99.9% of availability since it is crucial both for managing the charging points (and so, for the charging sessions) and for handling reservations from the eMSPs. In addition, having an unavailable CPMS would provoke economic issues for the CPOs, since its primary business is controlled by it.

3.5.3. Security

All communications between different entities must be on a secure channel and must be encrypted in order to avoid packet sniffing and spoofing. All the data provided by the users must be protected and their passwords must be not stored in clear.

3.5.4. Maintainability

The system must be designed in a way that it should be easily extended with additional features with minimum effort. To do so, both the CPMS and the eMSP must be designed with scalable and reusable modules.

3.5.5. Portability

The eMSP must be available in every mobile operating system (IOS and Android). Regarding the CPMS, it should be used in every web browser, from every device.

4 | Formal Analysis Using Alloy

4.1. Alloy model

```
// SIGNATURES

abstract sig Boolean {}
one sig TRUE extends Boolean {}
one sig FALSE extends Boolean {}

sig Username {}
sig Password {}
sig EmailAddress {}

sig Position {
        latitude: one Int,
        longitude: one Int
} {
        latitude ≥ 0 and longitude ≥ 0
}

sig DateTime {
        timestamp: one Int
} {
        timestamp ≥ 0
}

sig User {
        username: one Username,
        password: one Password,
        email: one EmailAddress,
        stationResearches: set StationsResearch,
        reservations: set Reservation,
        vehicles: set Vehicle,
        notifications: set Notification,
        paymentMethods: set PaymentMethod,
        payments: set Payment
}

sig StationsResearch {
        timestamp: one DateTime,
        position: one Position,
        nearbyStations: set ChargingPoint,
        distanceRange: one Int
} {
```

```
        distanceRange > 0
}

sig Reservation {
        from: one DateTime,
        to: one DateTime,
        socket: one Socket,
        chargingSession: lone ChargingSession
} {
        from.timestamp < to.timestamp
}

sig ChargingSession {
        isFinished: one Boolean,
        energyConsumed: one Int,
        batteryStatusEstimation: one Int
} {
        energyConsumed ≥ 0 and
        batteryStatusEstimation ≥ 0
}

sig Vehicle {
        vin: one Int,
        model: one CarModel,
        battery: one VehicleBattery
} {
        vin ≥ 0
}

sig CarModel {}
sig VehicleBattery {
        status: one Int
} {
        status ≥ 0
}

abstract sig Notification {
        timestamp: one DateTime
}

sig ChargingEndedNotification extends Notification {
        chargingSession: one ChargingSession
} {
        chargingSession.isFinished = TRUE
}

sig ChargingSuggestion extends Notification {
        from: one DateTime,
        to: one DateTime,
        suggestedVehicle: one Vehicle,
        chargingPoint: one ChargingPoint
} {
        from.timestamp < to.timestamp and
        timestamp.timestamp ≤ from.timestamp
}
```

```
sig PaymentMethod {}
sig Payment {
        amount: one Int,
        paymentMethod: one PaymentMethod,
        CPOInfo: one CPOPaymentReceiver,
        chargingSession: one ChargingSession,
        status: one PaymentStatus
} {
        amount > 0
}

abstract sig PaymentStatus {}
one sig FAILED extends PaymentStatus {}
one sig SUCCESS extends PaymentStatus {}

sig CPO {
        chargingPoints: set ChargingPoint,
        paymentReceiverInfo: one CPOPaymentReceiver
}

sig CPOPaymentReceiver {}

sig ChargingPoint {
        sockets: set Socket,
        position: one Position,
        availableDSO: set DSOEnergySource,
        batteries: set ChargingStationBattery,
        currentEnergySource: set EnergySource,
        tariffs: set Tariff
} {
        #availableDSO > 0
        #sockets > 0
        #tariffs > 0
}

sig Tariff {
        unitPrice: one Int,
        socketType: one SocketType
} {
        unitPrice > 0
}

sig SpecialOffer extends Tariff {
        validFrom: one DateTime,
        expiration: one DateTime
} {
        validFrom.timestamp < expiration.timestamp
}

sig Socket {
        type: one SocketType,
        chargingProfile: one ChargingProfile,
        status: one SocketStatus,
        reservations: set Reservation
}
```

```
sig ChargingProfile {
        validFrom: one DateTime,
        validTo: one DateTime,
        schedules: set ChargingSchedulePeriod
} {
        validFrom.timestamp < validTo.timestamp and
        #schedules > 0
}

sig ChargingSchedulePeriod {
        startTime: one Int,
        powerLimit: one Int
} {
        powerLimit > 0 and startTime ≥ 0
}

abstract sig SocketType {}
one sig SLOW extends SocketType {}
one sig FAST extends SocketType {}
one sig RAPID extends SocketType {}

abstract sig SocketStatus {}
one sig AVAILABLE extends SocketStatus {}
one sig RESERVED extends SocketStatus {}
one sig NOT_AVAILABLE extends SocketStatus {}
one sig CHARGING extends SocketStatus {}

abstract sig EnergySource {
        maxEnergyFlow: one Int,
        actualEnergyFlow: one Int
} {
        maxEnergyFlow > 0 and actualEnergyFlow ≥ 0 and
        actualEnergyFlow ≤ maxEnergyFlow
}

sig DSOEnergySource extends EnergySource {
        dso: one DSO,
        energyOffer: set DSOEnergyOffer //history of offers
} {
        #energyOffer > 0
}

sig ChargingStationBattery extends EnergySource {
        capacity: one Int,
        storageStatus: one Int,
        batteryStatus: one BatteryStatus
} {
        capacity > 0 and storageStatus ≥ 0 and
        (batteryStatus ≠ EXPORT implies actualEnergyFlow = 0) and
        (actualEnergyFlow > 0 implies batteryStatus = EXPORT)
}

abstract sig BatteryStatus {}
one sig IMPORT extends BatteryStatus {}
one sig EXPORT extends BatteryStatus {}
one sig IDLE extends BatteryStatus {}
```

```
one sig UNAVAILABLE extends BatteryStatus {}

sig DSO {}

sig DSOEnergyOffer {
        price: one Int,
        startTime: one DateTime,
        endTime: one DateTime
} {
        price > 0 and
        startTime.timestamp < endTime.timestamp
}

-------------------------------------------------------------------------------
// FUNCTIONS

fun sockets [c: CPO] : set Socket {
        c.chargingPoints.sockets
}

fun payedSessions[u: User]: set ChargingSession {
        u.payments.chargingSession
}

-------------------------------------------------------------------------------
// UTILITY PREDICATES

pred reservationsOverlapping[r1: Reservation, r2: Reservation] {
        r1. from.timestamp ≥ r2.from.timestamp and r1.from.timestamp < r2.to.timestamp
}

pred offersOverlapping[eo1, eo2: DSOEnergyOffer] {
        eo1.startTime.timestamp ≥ eo2.startTime.timestamp and eo1.startTime.timestamp <
            ↪ eo2.endTime.timestamp
}

// For simplicity, the range of a research here is a square and not a circle
pred nearRange[sr: StationsResearch, cp: ChargingPoint] {
        ((gte[cp.position.latitude, sr.position.latitude] and
        lte[sub[cp.position.latitude, sr.position.latitude], sr.distanceRange]) or
        (lt[cp.position.latitude, sr.position.latitude] and
        lte[sub[sr.position.latitude, cp.position.latitude], sr.distanceRange])) and
        ((gte[cp.position.longitude, sr.position.longitude] and
        lte[sub[cp.position.longitude, sr.position.longitude], sr.distanceRange]) or
        (lt[cp.position.longitude, sr.position.longitude] and
        lte[sub[sr.position.longitude, cp.position.longitude], sr.distanceRange]))
}

pred isActive [r: Reservation] {
        no r.chargingSession or r.chargingSession.isFinished = FALSE
}

-------------------------------------------------------------------------------
// FACTS

-- User
```

```
fact uniqueUsername {
        no disj u1, u2: User | u1.username = u2.username
}

fact uniqueEmail {
        no disj u1, u2: User | u1.email = u2.email
}

fact allPasswordsHaveUser {
        all p: Password | (some u: User | u.password = p)
}

fact allEmailAddressHaveUser {
        all e: EmailAddress | (some u: User | u.email = e)
}

fact allUsernameHaveUser {
        all usn: Username | (some u: User | u.username = usn)
}

-- Reservations

fact noOverlappingReservationsOfUser {
        no disj r1, r2: Reservation | reservationsOverlapping[r1, r2] and
                        (some u: User | r1 in u.reservations and r2 in u.reservations)
}

fact allReservationsHaveExactlyOneUser {
        all r: Reservation |
                (no disj u1, u2: User | r in u1.reservations and r in u2.reservations)
                    ↪ and (some usr: User | r in usr.reservations)
}

fact reservationSocketConsistency {
        all r: Reservation, s: Socket | r.socket = s implies
                (r in s.reservations and (no r.chargingSession implies s.status =
                    ↪ RESERVED) and
                (r.chargingSession.isFinished = FALSE implies s.status = CHARGING))
        all s: Socket |
                (all r: Reservation | not r in s.reservations or r.chargingSession.
                    ↪ isFinished = TRUE)
                iff (s.status = AVAILABLE or s.status = NOT_AVAILABLE)
        no disj r1, r2: Reservation |
                reservationsOverlapping[r1, r2] and r1.socket = r2.socket
        all s: Socket, r: Reservation | (r.socket = s and no r.chargingSession) implies
                (all r2: Reservation | r2 in s.reservations and (r2 = r or
                        (r2.chargingSession.isFinished = TRUE and r2.to.timestamp < r.
                            ↪ from.timestamp)))
        all r: Reservation | no disj s1, s2: Socket |
                r in s1.reservations and r in s2.reservations
}

//The system shall prevent the user to make new reservations if there are unsolved
    ↪ payment.
fact reservationNoUnsolvedPayment {
```

```
        all u: User |
                (some r: Reservation | no r.chargingSession and r in u.reservations)
                implies (all cs: ChargingSession | cs in u.reservations.chargingSession
                        implies (cs.isFinished = TRUE and
                                (one p: Payment | p in u.payments and p.chargingSession =
                                    ↪ cs and p.status = SUCCESS)))
        //only the last reservation can be unsolved
        all u: User, r: Reservation |
                (r in u.reservations and r.chargingSession.isFinished  = TRUE and
                no p: Payment | p.status = SUCCESS and p.chargingSession = r.
                    ↪ chargingSession)
                implies (no r2: Reservation |
                        (r2 in u.reservations and r2 ≠ r and r2.to.timestamp ≤ r.from.
                            ↪ timestamp
                        and (no p: Payment | p.status = SUCCESS and p.chargingSession =
                            ↪ r2.chargingSession)))
}

//The system shall allow the user to have only one reservation active at the same time.
fact onlyOneActiveReservation {
        all u: User | (no disj r1, r2: Reservation | r1 in u.reservations and
                r2 in u.reservations and isActive[r1] and isActive[r2])
}

fact onlyTheLastReservationCanBeActiveOrUnsolved {
        all u: User, r: Reservation |
                (r in u.reservations and isActive[r]) implies
                        (all r2: Reservation | (r2 in u.reservations and r2 ≠ r) implies
                                (not isActive[r2] and r2.to.timestamp ≤ r.from.timestamp)
                                    ↪ )
}

-- Stations Researches

fact allStationsResearchesHaveUser {
        all sr: StationsResearch | (one u: User | sr in u.stationResearches)
}

fact stationsResarchesNearbyStations {
        all sr: StationsResearch, cp: ChargingPoint |
                nearRange[sr, cp] iff cp in sr.nearbyStations
}

-- Charging Sessions

fact atMostOneChargingSessionPerReservation {
        all cs: ChargingSession | one r: Reservation | r.chargingSession = cs
}

//Not all charging sessions must have a notification since the user can terminate it
    ↪ earlier
//But there is at most one notification per charging session
fact oneNotificationPerChargingSession {
        no disj n1, n2: ChargingEndedNotification | n1.chargingSession = n2.
            ↪ chargingSession
}
```

```
fact allChargingSessionHaveOnePaymentAndAtMostOneSuccessful {
        all cs: ChargingSession | cs.isFinished = TRUE iff
                (some p: Payment | p.chargingSession = cs)
        no disj p1, p2: Payment | p1.chargingSession = p2.chargingSession and
                p1.status = SUCCESS and p2.status = SUCCESS
}

-- Payments

fact paymentReservationSameUser {
        all u: User, pm: Payment, r: Reservation |
                (pm in u.payments and pm.chargingSession = r.chargingSession) implies r
                    ↪ in u.reservations
}

fact paymentChargingSessionSameCPO {
        all cpo: CPO, pm: Payment | cpo.paymentReceiverInfo = pm.CPOInfo implies
                (one r: Reservation | r.chargingSession = pm.chargingSession and r.socket
                    ↪  in cpo.sockets)
}

fact allPaymentsHaveOneUser {
        all p: Payment | (one u: User | p in u.payments)
}

fact allPaymentMethodHaveUserOrPayment {
        all pm: PaymentMethod |  (some u: User | pm in u.paymentMethods) or
                        (some p: Payment | p.paymentMethod = pm)
}

-- Vehicles

fact allVehicleHaveUser {
        all v: Vehicle | (some u: User | v in u.vehicles)
}

fact allCarModelHaveVehicle {
        all cm: CarModel | (some v: Vehicle | v.model = cm)
}

fact allVehicleBatteriesHaveOnlyOneVehicle {
        all vb: VehicleBattery | (one v: Vehicle | v.battery = vb)
}

fact uniqueVIN {
        no disj v1, v2: Vehicle | v1.vin = v2.vin
}

-- Notifications

fact allNotificationsHaveUser {
        all n: Notification | (one u: User | n in u.notifications)
}

fact noSuggestionsAtTheSameTimeForTheSameUser {
```

```
        all cs1, cs2: ChargingSuggestion, u: User |
                (cs1 in u.notifications and cs2 in u.notifications and cs1 ≠ cs2) implies
                        cs1.timestamp.timestamp ≠ cs2.timestamp.timestamp
}

fact suggestionsVehicleOfUser {
        all usr: User, cs: ChargingSuggestion |
                (cs in usr.notifications) implies (cs.suggestedVehicle in usr.vehicles)
}

//The user must insert at least a vehicle in order to receive suggestions.
fact vehicleForSuggestions {
        all cs: ChargingSuggestion, u: User |
                cs in u.notifications implies #u.vehicles > 0
}

fact chargingEnededNotificationAfterChargingSessionIsFinished {
        all cn: ChargingEndedNotification, r: Reservation |
                cn.chargingSession = r.chargingSession implies cn.timestamp.timestamp ≥ r
                    ↪ .to.timestamp
}

fact chargingEndedNotificationChargingSessionSameUser {
        all u: User, cn: ChargingEndedNotification |
                cn in u.notifications implies cn.chargingSession in u.reservations.
                    ↪ chargingSession
}

-- Charging Points and CPOs

fact oneCPOPerPaymentReceiver {
        all pr: CPOPaymentReceiver | (one cpo: CPO | cpo.paymentReceiverInfo = pr)
}

fact CPOMustHaveAtLeastOneChargingPoint {
        all cpo: CPO | some cp: ChargingPoint | cp in cpo.chargingPoints
}

fact oneCPOPerChargingPoint {
        all s: ChargingPoint |
                (no disj c1, c2: CPO | s in c1.chargingPoints and s in c2.chargingPoints)
}

fact uniquePositionForChargingPoints {
        no disj s1, s2: ChargingPoint | s1.position = s2.position
        no disj p1, p2: Position | p1.latitude = p2.latitude and p1.longitude = p2.
            ↪ longitude
}

fact oneChargingStationPerSocket {
        all sock: Socket |
                (no disj p1, p2: ChargingPoint | sock in p1.sockets and sock in p2.
                    ↪ sockets)
}

fact allSocketsHaveChargingPoint {
```

```
        all sock: Socket | (some cp: ChargingPoint | sock in cp.sockets)
}

fact allChargingStationsHaveCPO {
        all cp: ChargingPoint | (one cpo: CPO | cp in cpo.chargingPoints)
}

-- Charging Profiles

fact allChargingSchedulePeriodHaveChargingProfile {
        all csp: ChargingSchedulePeriod |
                (some cp: ChargingProfile | csp in cp.schedules)
}

fact chargingSchedulePeriodStartingTimeConstraints {
        //no schedules at the same time
        all cp: ChargingProfile | no disj csp1, csp2: ChargingSchedulePeriod |
                csp1 in cp.schedules and csp2 in cp.schedules and csp1.startTime = csp2.
                    ↪ startTime
        //there always exists a schedule that starts at time 0
        all cp: ChargingProfile | one cs: ChargingSchedulePeriod |
                cs in cp.schedules and cs.startTime = 0
}

fact allChargingProfilesHaveSocket {
        all cp: ChargingProfile | (some sock: Socket | sock.chargingProfile = cp)
}

-- Energy Source

fact currentEnergySourceIsInBatteriesOrAvailableDSO {
        all cp: ChargingPoint, dso: DSOEnergySource |
                dso in cp.currentEnergySource implies dso in cp.availableDSO
        all cp: ChargingPoint, bat: ChargingStationBattery |
                bat in cp.currentEnergySource implies bat in cp.batteries
}

fact batteryCurrentEnergySource {
        all bat: ChargingStationBattery, cp: ChargingPoint |
                bat in cp.currentEnergySource implies bat.batteryStatus ≠ UNAVAILABLE
}

//A ChargingPoint can use only one DSO at a time as energy source
fact maxOneDSOInCurrentEnergySources {
        all cp: ChargingPoint | no disj dso1, dso2: DSOEnergySource |
                dso1 in cp.currentEnergySource and dso2 in cp.currentEnergySource
                    ↪
}

fact allBatteriesHaveExactlyOneChargingPoint {
        all bat: ChargingStationBattery | (one cp: ChargingPoint | bat in cp.batteries)
}

fact allDSOEnergySourceHaveChargingPoint {
        all des: DSOEnergySource | some cp: ChargingPoint | des in cp.availableDSO
}
```

```
fact chargingPointAtMostOneEnergySourceOfDSO {
        all cp: ChargingPoint |
                (no disj des1, des2: DSOEnergySource |
                        des1.dso = des2.dso and des1 in cp.availableDSO and des2 in cp.
                            ↪ availableDSO)
}

fact allDSOHaveAtLeastOneEnergySource {
        all ds: DSO | some des: DSOEnergySource | des.dso = ds
}

//While a DSO can serve multiple charging points, the same energy source cannot
//since they might have different actual energy flows
fact noSameDSOEnergySourceForDifferentChargingPoints {
        all esrc: DSOEnergySource | no disj cp1, cp2: ChargingPoint |
                esrc in cp1.availableDSO and esrc in cp2.availableDSO
}

//If energy source is not used than the actual energy flow is zero
fact ifDSOEnergySourceIsNotUsedThenEnergyFlowIsZero {
        all cp: ChargingPoint, esrc: DSOEnergySource |
                (esrc in cp.availableDSO and esrc ≠ cp.currentEnergySource)
                        implies esrc.actualEnergyFlow = 0
}

fact onlyOneDSOEnergySourcePerEnergyOffer {
        all eo: DSOEnergyOffer | one es: DSOEnergySource | eo in es.energyOffer
}

fact noTwoOffersAtTheSameTime {
        all es: DSOEnergySource | no disj eo1, eo2: DSOEnergyOffer |
                eo1 in es.energyOffer and eo2 in es.energyOffer and
                (offersOverlapping[eo1,eo2] or offersOverlapping[eo2,eo1])
}

-- Tariff

fact allTariffHaveOnlyOneChargingPoint {
        all tar: Tariff | (one cp: ChargingPoint | tar in cp.tariffs)
}

//All type of sockets in a charging point must have a tariff
fact socketTariff {
        all sock: Socket, cp: ChargingPoint |
                sock in cp.sockets implies (some t: Tariff | t in cp.tariffs and t.
                    ↪ socketType = sock.type)
}

//There shouldn't be two base tariffs on the same type of sockets (but there can
//some special offers and a tariffs on the same socket)
fact onlyOneBaseTariffsOnTheSameSocketType {
        all cp: ChargingPoint | #(cp.tariffs - SpecialOffer) = #cp.sockets.type
}

--------------------------------------------------------------------------------
```

```
//PREDICATES FOR DYNAMIC MODELLING

pred makeReservation[u, u': User, r: Reservation] {
        //precondition
        no r.chargingSession

        //postconditions
        u'.reservations = u.reservations + r
        u'.username = u.username
        u'.password = u.password
        u'.email = u.email
        u'.stationResearches = u.stationResearches
        u'.vehicles = u.vehicles
        u'.notifications = u.notifications
        u'.paymentMethods = u.paymentMethods
        u'.payments = u.payments
}
run makeReservation for 4 but exactly 2 Reservation, 1 User

pred addStationsResearch[u, u': User, sr: StationsResearch] {
        u'.reservations = u.reservations
        u'.username = u.username
        u'.password = u.password
        u'.email = u.email
        u'.stationResearches = u.stationResearches + sr
        u'.vehicles = u.vehicles
        u'.notifications = u.notifications
        u'.paymentMethods = u.paymentMethods
        u'.payments = u.payments
}
run addStationsResearch for 4 but exactly 2 ChargingPoint

pred payChargingSession[u, u': User, cs: ChargingSession, p: Payment] {
        //preconditions
        p.chargingSession = cs
        one r: Reservation | r in u.reservations and r.chargingSession = cs
        no p1: Payment | p1 in u.payments and p1.chargingSession = cs and p1 ≠ p and p1.
            ↪ status = SUCCESS

        //postconditions
        u'.reservations = u.reservations
        u'.username = u.username
        u'.password = u.password
        u'.email = u.email
        u'.stationResearches = u.stationResearches
        u'.vehicles = u.vehicles
        u'.notifications = u.notifications
        u'.paymentMethods = u.paymentMethods
        u'.payments = u.payments + p
}
run payChargingSession

pred setChargingPointDSO [cp, cp': ChargingPoint, dso: DSOEnergySource] {
        //precondition
        dso in cp.availableDSO
```

```
        //postconditions
        cp'.sockets = cp.sockets
        cp'.position = cp.position
        cp'.availableDSO = cp.availableDSO
        cp'.batteries = cp.batteries
        cp'.tariffs = cp.tariffs
        cp'.currentEnergySource = dso
}
run setChargingPointDSO for 4 but exactly 3 DSOEnergySource, 1 ChargingPoint

pred addSpecialOfferToChargingPoint [cp, cp': ChargingPoint, s: SpecialOffer] {
        //preconditions
        some sock: Socket | sock in cp.sockets and sock.type = s.socketType

        //postconditions
        cp'.sockets = cp.sockets
        cp'.position = cp.position
        cp'.availableDSO = cp.availableDSO
        cp'.batteries = cp.batteries
        cp'.tariffs = cp.tariffs + s
        cp'.currentEnergySource = cp.currentEnergySource
}
run addSpecialOfferToChargingPoint

-------------------------------------------------------------------------------
// ASSERTIONS

//Goal: allow users to visualize nearby charging stations
assert allowStationsResearches {
        all sr: StationsResearch | (some u: User | sr in u.stationResearches) and
                ((some cp: ChargingPoint | nearRange[sr, cp]) implies #sr.nearbyStations
                    ↪ > 0)
}
check allowStationsResearches for 4

// Goal: allow users to book a charge
// There might be some reservations that do not have a charging session
assert bookCharge {
        all r: Reservation |
                no r.chargingSession implies (one u: User | r in u.reservations)
}
check bookCharge for 4

// Goal: allow a user to start a charging session
assert startChargingSession {
        all cs: ChargingSession |
                (some u: User, r: Reservation | r in u.reservations and r.chargingSession
                    ↪  = cs)
}
check startChargingSession for 4

// Goal: allow users to pay for the service
assert payForService {
        all p: Payment | p.chargingSession.isFinished = TRUE and
                (one u: User | p in u.payments and p.chargingSession in u.reservations.
                    ↪ chargingSession)
```

```
}
check payForService for 4

//Goal: allow users to receive suggestions
assert userSuggestions {
        all cs: ChargingSuggestion | (some u: User | cs in u.notifications)
}
check userSuggestions for 4

//This assertion checks if the model for the reservations is correct
assert testConsistency {
        not (#User = 1 and #ChargingSession = 2 and #Reservation = 4 and #Payment = 4)
}
check testConsistency for 6

-------------------------------------------------------------------------------
// WORLDS

//General world with multiple Users and CPOs
pred world1 {
        #User = 2
        #CPO = 2
        #ChargingPoint ≥ 3
        #Reservation ≥ 4
        #Socket ≥ 3
        some s: Socket | s.type = SLOW
        some s: Socket | s.type = FAST
        some s: Socket | s.type = RAPID
        #ChargingSession > 2
        some cs: ChargingSession | cs.isFinished = TRUE
        #Payment > 1
        some p: Payment | p.status = SUCCESS
        some p: Payment | p.status = FAILED
}
run world1 for 4

//This world is focused on charging suggestions
pred world2 {
        #Vehicle > 2
        #ChargingSuggestion > 2
        #User = 2
        #CPO = 1
        #ChargingPoint > 1
        #Reservation = 0
}
run world2 for 6

//This world is focused on charging points
pred world3 {
        #User = 0
        #CPO ≥ 2
        #ChargingStationBattery > 0
        #DSOEnergySource > 3
        #Tariff ≥ 4
        #SpecialOffer > 0
        some s: Socket | s.type = SLOW
```

```
        some s: Socket | s.type = FAST
        some s: Socket | s.type = RAPID
        some cp: ChargingPoint | #cp.sockets > 1
        #ChargingProfile = #Socket
        #ChargingSchedulePeriod > #ChargingProfile
        some cp: ChargingPoint | #cp.currentEnergySource > 1
        some dso: DSOEnergySource | #dso.energyOffer > 1
}
run world3 for 8

//This world is focused on the user registration
pred world4 {
        #Vehicle > 3
        #User > 2
        #CPO = 0
        some u: User | #u.vehicles > 1
        some u: User | #u.paymentMethods > 1
        #DateTime = 0
        #Position = 0
}
run world4 for 4

//This world is focused on reservations on the same socket
pred world5 {
        #Socket = 1
        #User ≥ 2
        #Reservation ≥ 3
        some r: Reservation | no r.chargingSession
        #Payment ≥ 2
        #ChargingEndedNotification > 0
        some p1, p2: Payment | p1 ≠ p2 and p1.chargingSession = p2.chargingSession
                and p1.status ≠ p2.status
}
run world5 for 6

//This world is focused on the stations researches
pred world6 {
        #User > 0
        #StationsResearch = 2
        #CPO = 1
        #DSO > 1
        #ChargingPoint ≥ 4
        all cp: ChargingPoint | #cp.sockets = 1
        #Reservation = 0
        all sr: StationsResearch | sr.distanceRange ≤ 3
}
run world6 for 6
```

Figure 4.1: Result of the execution of "Execute all".

4.1.1. First world

The first world (Figure 4.2) gives a general view of the domain, where there are multiple Users and CPOs. Each CPO can own multiple Charging Points and each of them can have multiple Sockets.

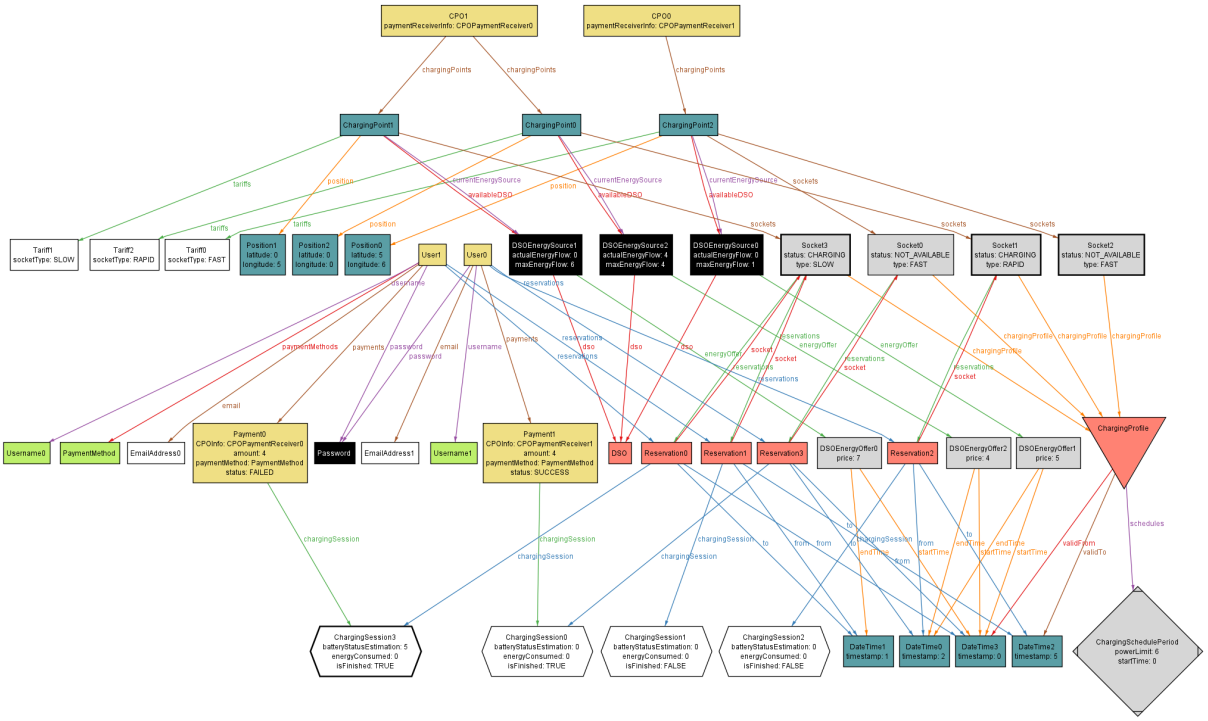

Figure 4.2: World 1

4.1.2. Second world

The second world (Figure 4.3) is focused on **charging suggestions**. This model shows that a charging suggestion belongs only to a user and contains a suggested vehicle that is registered inside the vehicles of the user. Each charging suggestion is suggesting the user go and charge the suggested vehicle at a certain charging point. Each user cannot receive more than one suggestion at each time frame.

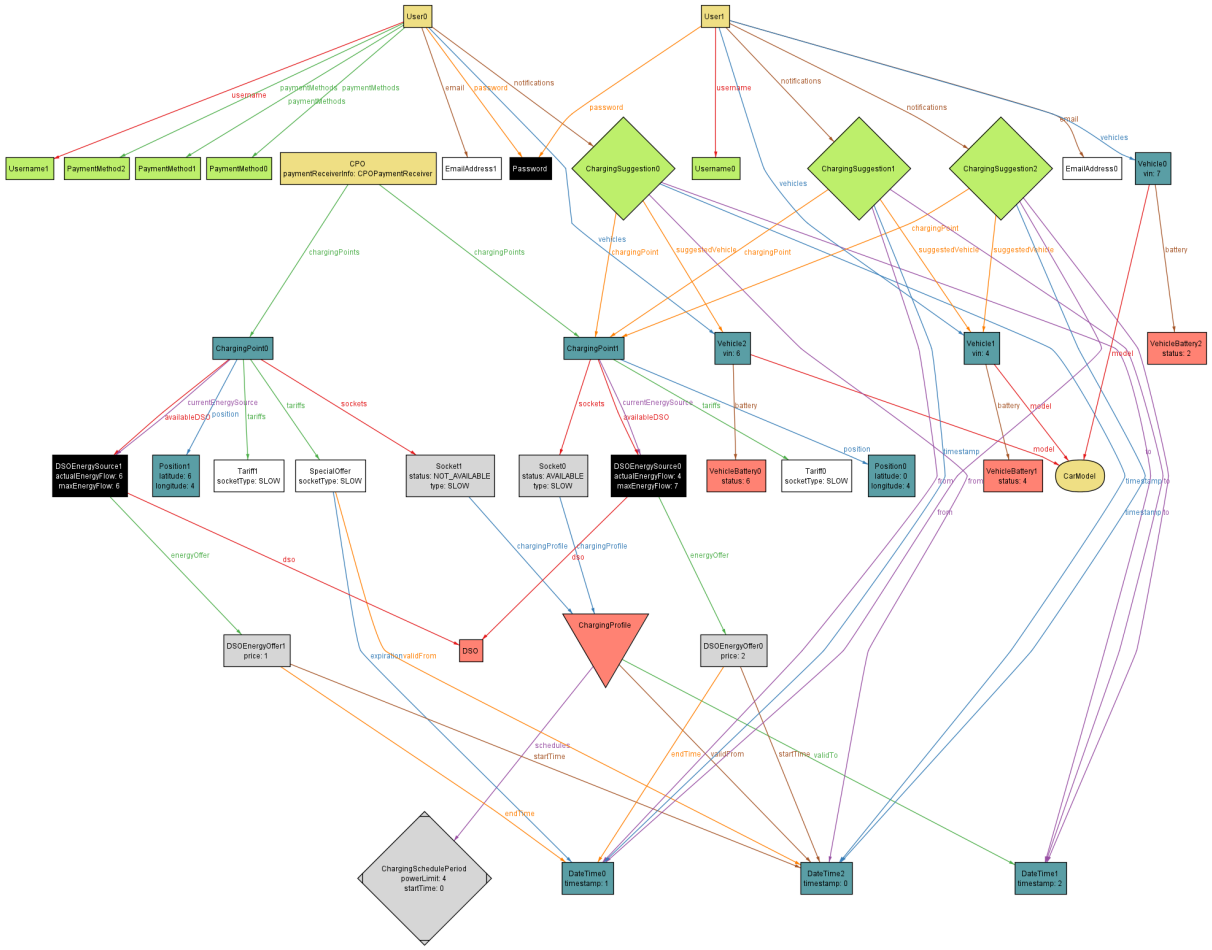

Figure 4.3: World 2

4.1.3. Third world

In the third world (Figure 4.4) the focus is on the structure of a **charging point**. Each charging point can have multiple sockets of different types (SLOW, FAST, RAPID). If a certain charging point has at least a socket of a certain type, it must contain a tariff for that type of socket. A charging point can have also some special offers for some types of sockets, and these are valid for a certain period of time. Each charging point can be related to multiple energy sources, these can be batteries or can come from a DSO. Since a DSO can provide different offers for different CPOs (in terms of both the price and the maximum energy flow), each DSO has a different energy source for different CPOs. These energy sources contain also the history of all the offers that the DSO have made. The charging point can contain multiple current energy sources, but only one of those can be an energy source of a DSO.

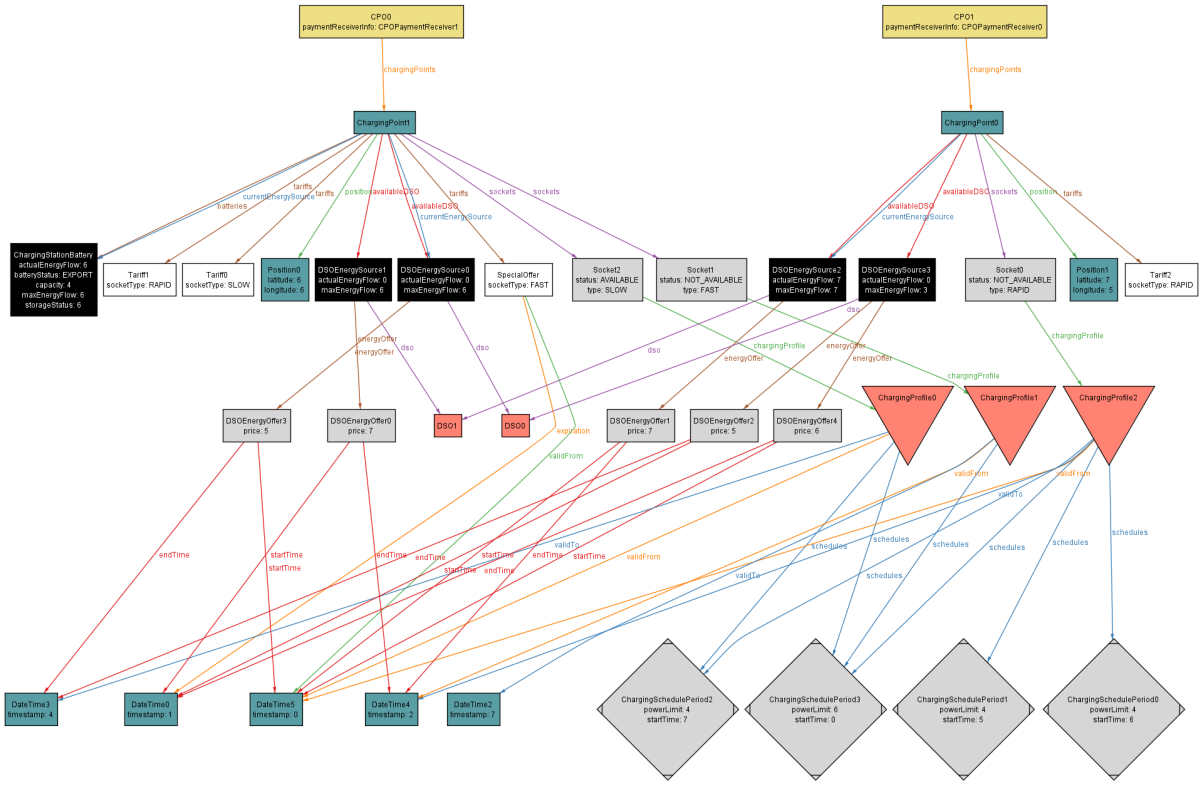

Figure 4.4: World 3

4.1.4. Fourth world

The fourth world (Figure 4.5) is concentrated on the users' data. Each **user** can register some vehicles and each vehicle can be registered by more than one user.

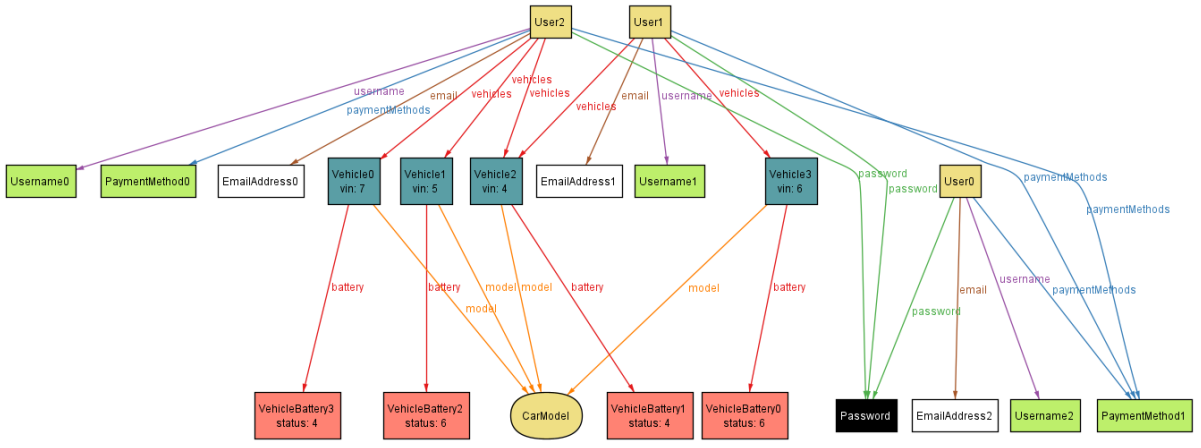

Figure 4.5: World 4

4.1.5. Fifth world

The fifth world (Figure 4.6) is focused on the **registrations** on the same socket. Two registrations cannot overlap if they are related to the same socket. Each user can make a reservation only if he doesn't have any pending payments (i.e. if all the payments have SUCCESS as a status) and only his last reservation can be active (i.e. with a not started or not finished charging session).

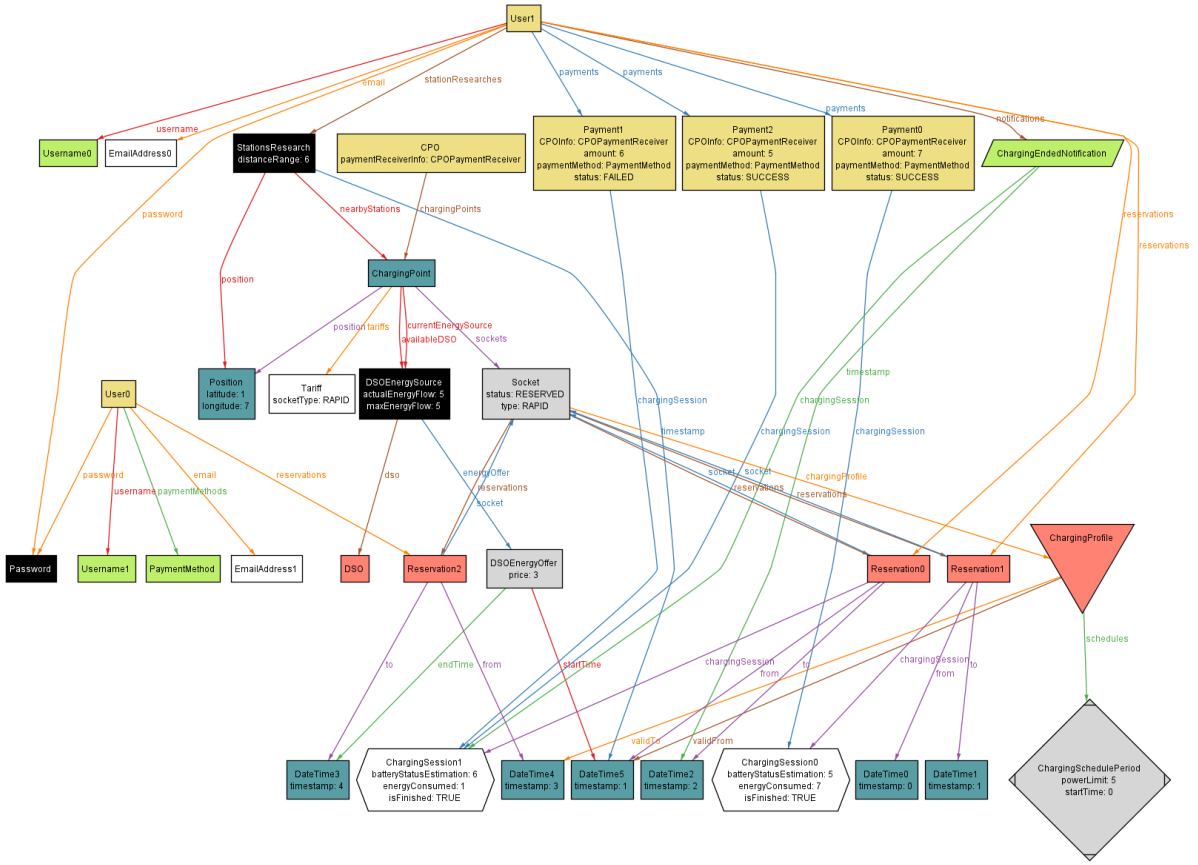

Figure 4.6: World 5

4.1.6. Sixth world

In the sixth world (Figure 4.7) the focus is on the **stations researches**. A user is connected to some stations' research; each of those has a position around which the research is made and a distance range. All charging points that are inside the distance range are returned.

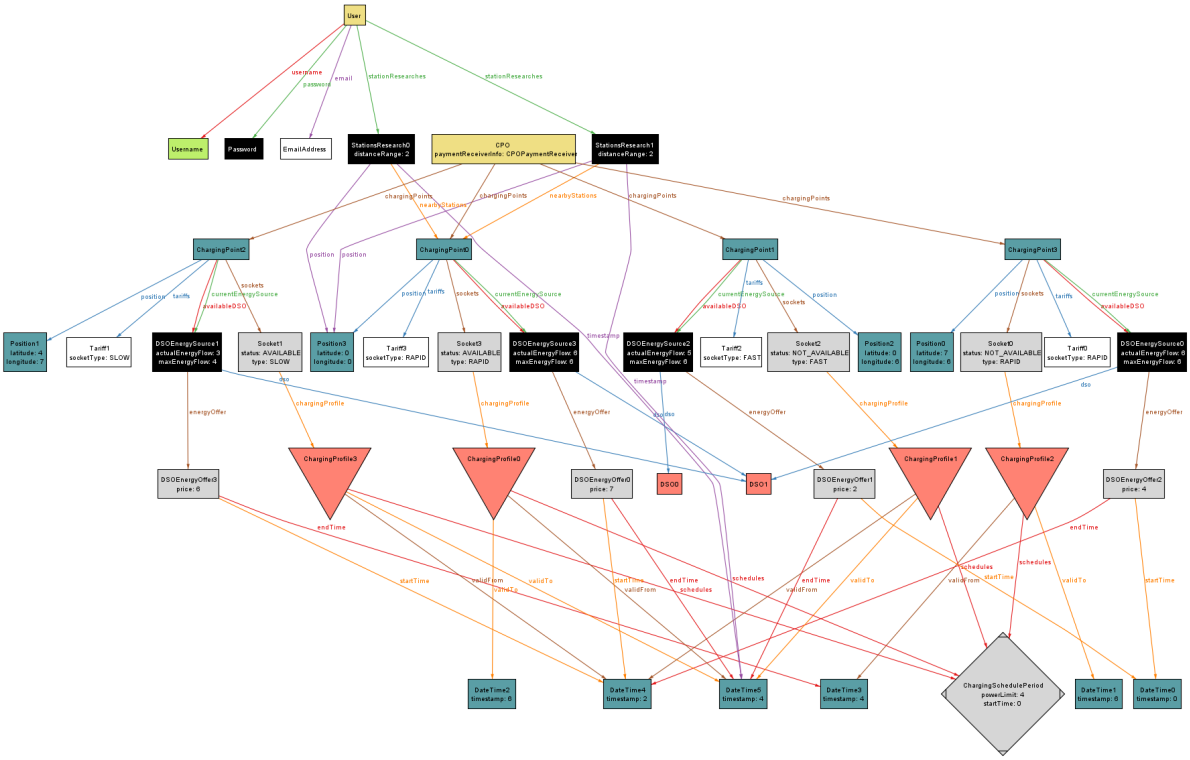

Figure 4.7: World 6

5 | Effort Spent

5.1. Lorenzo Ferretti

Task	Hours Spent
Introduction	5
Overall Description	20
Specific Requirements	25
Formal Analysis	5

5.2. Lorenzo Manoni

Task	Hours Spent
Introduction	10
Overall Description	15
Specific Requirements	25
Formal Analysis	5

5.3. Carlo Sgaravatti

Task	Hours Spent
Introduction	8
Overall Description	10
Specific Requirements	15
Formal Analysis	22