Politecnico di Milano

eMall – e-Mobility for All

Requirements Analysis and Specification Document

Beliakov Maksim
10913275

Aliyev Rustam
10897101

Yizhou Wu
10926468

January 8, 2023

POLITECNICO
MILANO 1863

Contents

1. Introduction

1.1 Purpose

However, in order to widely spread the use of electrical vehicles among the population, the corresponding servicing infrastructure is required. First, charging points infrastructure is to be developed so that the users of electrical mobility are able to conveniently charge their cars. Therefore, the corresponding managing system (eMSP) for this infrastructure should be developed which must take care of providing the end user with the information about charging points nearby and their features, managing the charging actions such as booking, starting, finishing, payment of charging. Also, this managing system interacts with the charging point managing system (CPMS) which is responsible for setting the status of the point, charging a vehicle, interaction with an energy distribution system operator (DSOs).

1.1.1 Goals

Goal	Description
G1(eMSP)	User should know the locations of charging stations nearby, their costs and special offers.
G2(eMSP)	The user should be suggested to go and charge the car battery based on user's schedule, car battery level, special offers and availablity of socket at nearby charging stations
G3(eMSP)	The user should be able to book a charge at a charging station
G4(eMSP)	The user should be able to manage a charging process in a charging station i.e. start charging, paying, getting a notification of the charging end
G5(eMSP)	The user should be able to use the functionalities of the system based on provided personal, the data about the cars, payment details
G1(CPMS)	The CPO and its operators should be able to decide for each Charging Point from which DSO to acquire energy manually or automatically basing on chosen conditions
G2(CPMS)	The CPO and its operators should be able to decide for each Charging Point whether to store or not energy in batteries manually or automatically basing on chosen condition satisfied
G3(CPMS)	The CPO and its operators should be able to for each Charging Point the source of energy used for charging in a CP manually or automatically basing on chosen conditions
G4(CPMS)	The charging socket should charge only electric car that actually booked a recharge
G5(CPMS)	The CPO and its operators should be able to know all details about charging station in real-time
G6(CPMS)	The charging station should be able to receive and handle booking and charging request automatically

1.2 Scope

This document concerns all the features provided by the eMSP system and the CPMS system, and the interactions between them.

1.2.1 World Phenomena

Identifier	Description
WP1	Driver's car is running out of battery
WP2	Driversets up the schedule of a day on his phone
WP3	Driverhas an occupation for a certain timeframe
WP4	CPO acquire energy from different DSOs
WP5	DSOs energy cost changes over time
WP6	Charging point serve only cars with booking
WP7	Charging point has limited number of charging socket
WP8	Charging point can have batteries that can store and provide energy
WP9	CPO evaluate ability of its charging points at earning money

1.2.2 Shared phenomena

Identifier	Description
SP1	User opens application and see the charging points nearby, their offers and costs
SP2	Application accesses the schedule of a user in phone
SP3	User books a charge through application
SP4	Application notifies user that the charging is finished
SP5	User starts charging through eMSP application
SP6	User enters the data about his car to application
SP7	CPO delegates the decision from which DSO to acquire energy
SP8	CPMS decides to charge batteries based on current DSO prices and amount of energy left in batteries
SP9	DSO offer energy at low cost at specific charging points
SP10	CPMS notifies eMSP that this specific charging point has energy at low price based on amount of energy in batteries
SP11	eMSP evaluates the amount of energy left in car
SP12	eMSP inquires the nearest charging points about number of chargers and time slots they are available
SP13	CPMS monitors the amount of energy left in batteries at this specific charging point

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Definition	Description
Special offer	Offer w.r.t user is used as a general term for any kind of commercial propositions e.g. discounts.
Application	The mobile application through which user gets access to features provided by eMSP

1.3.2 Abbreviations

Abbreviation	Description
RASD	Requirements Analysis and Specification Document
WP	World Phenomena
SP	Shared Phenomena
eMSP	Electric-mobility service provider
CPMS	Charging Point Management System
CP	Charging point (station)
CPO	Charging Point Operator
DSO	Distribution System Operator
GX	Goal number X
DX	Domain assumption number X
RX	Requirement number X

1.4 Revision History

Version	Date	Notes
0.1	2022/12/21	First parts of documents
0.2	2022/12/23	Integrated second parts.
1.0	2022/12/23	Refined the document and first release
2.0	2022/12/30	Added missing scenario of setting price and special offer in CPMS and minor changes. (read README file for details)
2.1	2023/01/08	Document refinement

1.5 Reference Document

- The specification document "01. Assignment RDD AY 2022-2023_v3"

1.6 Document Structure

This document is structured in the following way:

1. The first chapter is an introduction and overview of the project, setting the context that led to its development, the goals to be achieved, and a general description of its functionality.
2. The second chapter is a formal description of the domain model and the project through the extensive use of class diagrams and state machine diagrams. The class diagram provides a high-level description of the domain entities and their relationships, while the state machine diagram focuses on modeling the most important entities through their state transitions. All functional requirements and domain assumptions are also presented here to achieve the previously stated goals.
3. The third chapter presents the non-functional requirements, which are deepened thanks to the description of possible use cases using natural language and sequence or activity diagrams, and illustrates the design constraints.
4. The fourth and final chapter presents a formal analysis of the model through the use of the open source Alloy language and tool. Some of the configurations created by the tool are included.
5. In section five there is a presentation of the project members total effort spent.
6. Section six contains the references used

2. Overall description

2.1 Product perspective

2.1.1 Scenarios

1. **Car driver wants to start using the eMSP**

   The owner of an electric car John wants to use charging points available for eMSP users (**Note** : charging points belong to CPOs, eMSP's main function should be an aggregator of services provided by CPOs through their CPMS) so that he can use benefits offered by this infrastructure such as knowing about nearby charging points, their availabilities and costs, booking the charging points etc. John opens the application, goes through registration procedure and gets access to the e-mobility service.

2. **Car driver books a charge at the nearest available charging point**

   The electric car owner Bob while driving noticed that his car is running out of energy. He launches the eMSP mobile application, sees the map where his location and locations of the charging points nearby are visible. Bob decides which charging point is better located according to his route. He clicks on the icon of this charging point on the map. Then eMSP shows the information: if special offers are available, number of charging sockets available, their type such as slow/fast/rapid, their cost, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed. Then he books the charging of the desired type at this charging point for a suitable timeslot.

3. **CPO achieves digital transformation of its business process**

   CPO company Inil Y owns many charging stations, handling the booking via company's call center and dealing with drivers, who arrive to charging station willing to charge their cars without a proper booking is always creating so many conflict and overlapping on usage of charging sockets, because of unavoidable communication delay between call center department and field operation in actual charging stations. Thus Inil Y decided to digitalize its business processes, basing them entirely on IT infrastracture, to have the booking and charging request of every charging points more smoothly and clearly executed relying on a Charging Points Managment System.

4. **Helping CPO to monitor the status of its charging points**

   CPO company Oxigenity owns multiple charging points across country, it would like to visualize each single charging points's working status at real-time, instead of sending operators around to gather their data one by one and waiting their report. So the company decided to take the approach of CPMS, installing sensors over charging stations, doing the work once for ever. Now they can keep track of all detailed information their assets (charging points, charging sockets, batteries) by some simple clicks on computer to access CPMS without necessity of sending resources on the frontline.

5. **CPO automatize the application of energy acquiring and usage strategies.**

   In this continuously varying energy market, the speed to change the strategies of using and acquiring the energy is an essential factor to maximize company's profit. CPO company Slowned elaborated an sophisticated mathematical model able to determine the best strategy for its charging stations, regarding whether to spend energy stored in its batteries to satisfy the charging needs or to acquire energy from DSO, and when is the best moment to store energy in batteries. To guarantee that these strategies are contextually elaborated and executed, Slowned decided to rely on CPMS exploiting the computational ability of modern IT infractracture, all it needs to do now is simply to convert the model in CPMS accepted conditions and configure the system to automatically handle the acquiring and usage of the energy in charging stations.

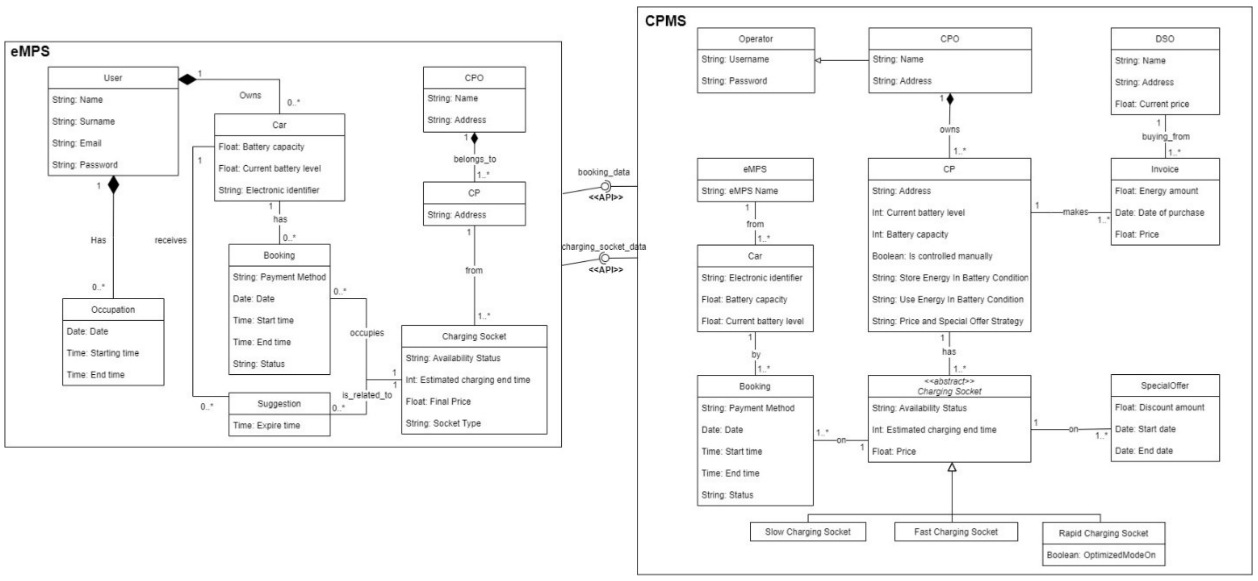
6. **CPO inspecting the cars under charge in a charging point**

   A CPO company needs frequently to assign Inspection task to its operators, requiring them to go around the charging points, checking that there is no fraud scene happenning, for instance, the car that is getting charged does not correspond to the car that booked the charging. Previously company should spend a not negligible amount of human force in order to elaborate manually the list of booking of charging points in last hour when operators arrive to places, and transfer it to them. Now thanks to the recent introduction of CPMS, this kind of inspection task is no longer necessary, as CPMS guarantees also that the cars under charge are the ones booked, leaving no space for fraud situation anymore.

7. **Car driver pays for the charging** Electric car driver Matteo has just got his car charged. But he's in a hurry, so he doesn't have time to stay in a line to pay for the obtained service with the cash deposit machine. Fortunately, there's an opportunity to pay right from his phone. Therefore he opens the application and pays for the service using a debit card attached to his eMSP account in seconds and gets back to his route.

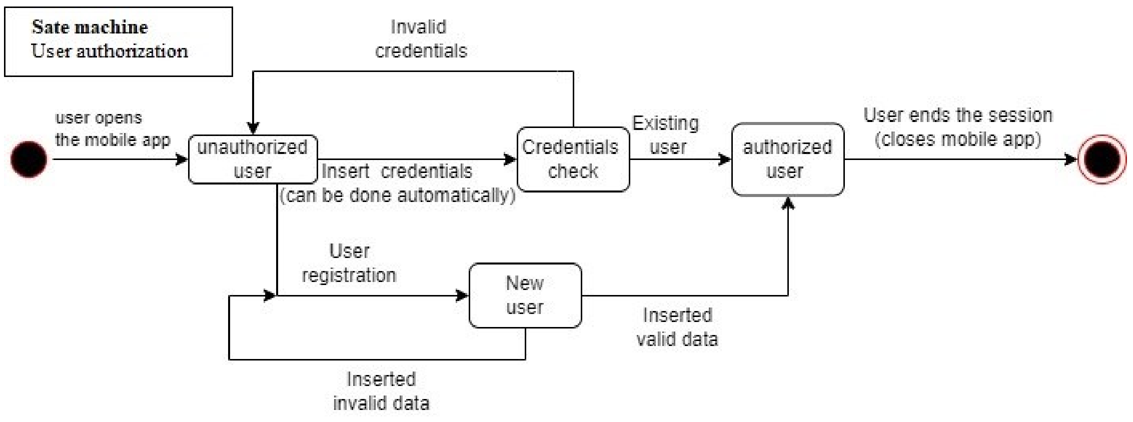
8. **Car driver receives a suggestion to go to a certain charging station and charge his car** An electric car driver Smit while enjoying the hitting the gas of his electric car gets a notification on his mobile phone that his car battery level is 40%, he doesn't have any occupation whithin the next 3 hours and there's a charging point nearby with available charging sockets of fast type, and moreover this charging point offers a discount for charging of more than 50% wthin the next 5 hours. So, he is suggested use the opportunity of charging. Smit finds this suggestion reasonable and books a charge with the abovementioned charging socket.
9. **Car driver starts charging of a car** An electric car driver Andrea after he had booked a charge at the appropriate charging point for him just arrived to the charging point and wants to start charging his car. He plugs the charger into the car and after opens the application and pushes the button "start charging" and the charging starts. When the charging is finished he receives the notification on his mobile.

2.1.2 Class diagram

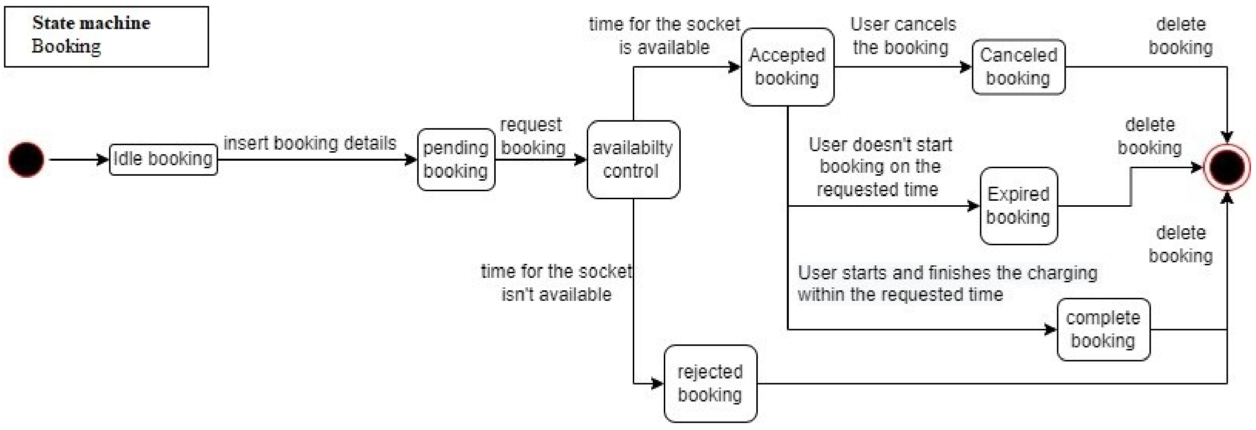


2.1.3 State charts

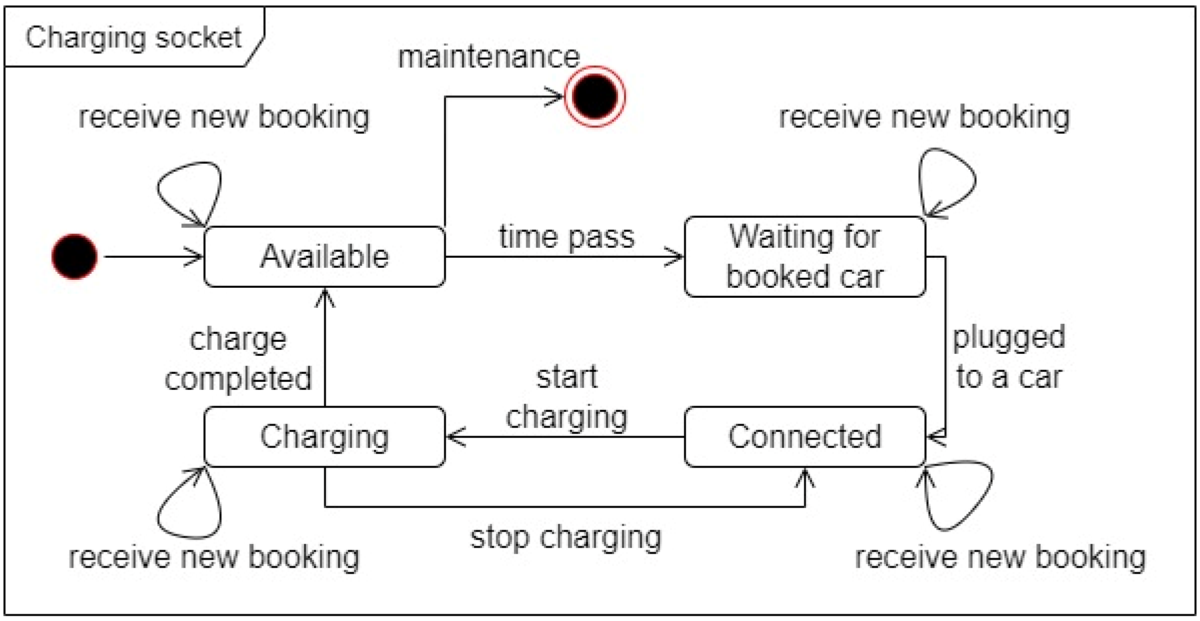
State diagram of User authorization



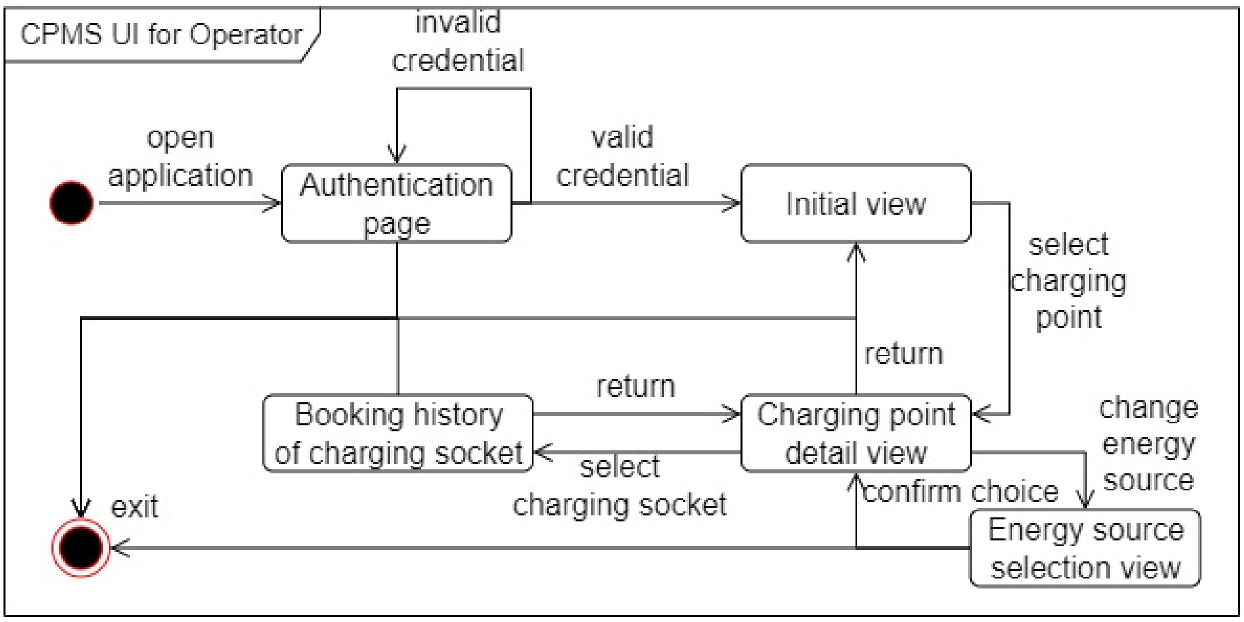
State diagram of booking (from eMSP side)



State diagram of charging socket



State diagram of CPMS User Interface



2.2 Product Functions

This section describes the main functionalities of eMSP and CPMS in much deeper and more comprehensive way.

2.2.1 Providing elecric car drivers with the ability of convenient managing the charging process

The main goal of the eMSP system is to provide the electric car drivers with the convenient way of getting their car charged so that it introduces minimal interference and constraints on driver's daily schedule. To achieve this the system allows several features to drivers such as knowing about the charging stations nearby, their cost, any special offer they have, booking a charge in a specific charging station for a certain timeframe, starting the charging process at a certain station, notifying the user when the charging process is finished, paying for the obtained service.

Apart from these ones, eMSP system does proactive suggestions to a driver to go charge the vehicle based on the level of battery energy, the schedule of the driver, the special offers made available by some CPOs, and the availability of charging slots at the identified stations.

2.2.2 Helping CPO at management of charging points and basic business process

One of the most important contributions of a CPMS system is to digitalize the core of a CPO company's business model, by exposing the availability of every single charging socket in each charging points clearly and structurely through IT infrastructure to end-users, the potentiality of charging points could be totally exploited, in ideal case, all charging sockets could be occupied without any gap between usage, therefore maximazing the profit of comany.

Not only, CPMS could help CPO to monitor and eventually keep track of working status and performance of each charging station with real-time data, which are the foundamental of company's business analytical operations.

Thanks to approach of adopting uniform API while communicating with other node of IT infrastructure of Electric mobility industry, CPMS is also highly scalable and customizable, leaving freedom to various provider of system to implement their own features. For instance the possibility to set special offer on price of energy, to dynamically change the strategies of energy acquisition plans and so on, as long as the implementation follows the basilar standard interface between systems, it can interact with other nodes without any problem.

(Managing a charging point is a dificult task since there are many decisions which should be made. During the charging of a single car CPO needs tostart charging a vehicle according to the amount of power supplied by the socket, and monitor the charging process to infer when the battery is full, decide from which sourse to acquire the energy (which DSO, battery or a mix thereof), decide the cost of charging of each type based on the amount of energy in batteries and prices by DSO at the moment.

Moreover, special offers for drivers of cars of a certain type should be made based on the amount of energy in batteries and prices by DSO at the moment.

Any human being is unable to consider so many factors to make an optimal decision. So, the CPMS is designed to help.)

2.3 User characteristics

The following actors are considered in the system to be.

1. **Unregistered Electric car driver** A driver of an elecrtic car who wants to use the functionalities and convenience of the services which eMSP provides and thus needs to register himself/herself into the system to be.
2. **Electric car driver** A registered driver who is associated to one or several cars and uses the eMSP in order to be able to know about the charging stations, book a charge, manage the charging process. The driver uses eMSP through the mobile appliction.
3. **CPO Operator**

   A registered user on CPMS representing the employee of the owner CPO company, that use the system to monitor the status of Charging Points and to administrate the charging aspect and energy acquisition aspect of them.

4. **CPO**

   The unique (singleton) operator on CPMS representing the company itself, with all privileges of a normal CPO Operator, so everything accessible to CPO Operator is accessible also to CPO, plus the ability of inserting new Charging Points and configuring every aspect of them.

2.4 Assumptions, dependencies, and constraints

2.4.1 Domain assumptions

D1	One user can own several cars and insert the data about the ones he wants the app to use for
D2	User inserts the truthful data about himself/herself and his cars
D3	User has several options of payment (i.e. debit card, paypal, Cash)
D4	eMSP application has access to the user's schedule in the mobile phone
D5	Mobile app(eMSP) knows precisely about the current level of the car battery through some interfaces (e.g. bluetooth)
D6	Mobile app(eMSP) has access to the car navigation system
D7	Mobile app(eMSP) knows about the accurate location of the user's car
D8	The car is driven by the person associated to this car during the registration
D9	User triggers the charging process only when the car is connected to a charging socket
D10	User manually connects and disconnects the car from a charging socket
D11	Charging socket can infer if the battery is full when the power supply stops by means of a built-in controller
D12	Each charging points has cash deposit machines to provide the opportunity of the cash payment method
D13	Each car has an electronic identifier which can be accessed by eMSP and using which eMSP can verify that the user is driving the car he specified in the app
D14	CPMS provider gives the CPO account's credential on CPMS to CPO company
D15	CPO company will give the new profile's credential to its employee
D16	The status of batteries in a CP is monitored by sensors and transfered to CPMS

D17	The status of charging socket in a CP is monitored by sensors and transfered to CPMS
D18	The source of energy used in CP can be changed both remotely and manually
D19	Conventioned DSOs will provide updated information of their energy to CPMS through API
D20	Each charging socket in CP has an identifier observable in real world by car driver that matches with the one stored in CPMS
D21	The process of charging can be started and stopped by CPMS
D22	The CPMS subsystem shall be able to identify the fact whether the car is physically connected to a charging socket
D23	Payment for charging is handled by Third-party system which synchronize the payment result with eMSP and CPMS through uniform APIs.

3. Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface of *eMSP* in case of ordinary users (Clients who wish to charge their vehicles) is a mobile application. It should be available to as many people as possible, thus it should support even older versions of mobiles OSs (Android, Ios, Windows Phone if it still exists). In case of CPOs (*CPMS*) the user interface is a desktop application. It is obvious that specially trained workers will work there, thus it may not be as user-friendly as a mobile application, but it will definetly be more complex and harder to use.

3.1.2 Hardware Interfaces

eMSP is a lightweight mobile application, thus any smartphone (With Wi-Fi and bluetooth being turned on) should be able to run it. All the calculation work and other stuff is done on some server, this application simple receives the results and shows them to the client. *CPMS* is considered to be a desktop application, thus in order to launch it CPO will need a personal computer. Also, it is important to track the information of the CPs regarding the charging sockets, therefore there will be need in sensors with an access to Wi-Fi, in order to be able to exchange the information via APIs.

3.1.3 Communication Interfaces

Having 2 subsystems, *eMSP* and *CPMS*, it is important to implement the cooperation between these two. They cannot work as seperate entities, thus communication between these two and with some external sources is a key-point here.

It would be easier to list all the communications that occur in our system:

eMSP:

- Retrieveing data about geolocation, electronic identifier and battery level from the car via bluetooth
- Retrieveing data about available slots in CPs from CPMS

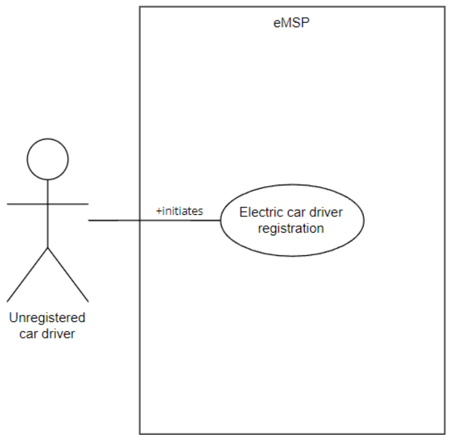
CPMS:

- Retrieveing data about the process of charging from the sensors in charging sockets in CPs
- Retrieveing data about the statuses of charging sockets from CPs
- Retrieveing data from DSOs regarding the available energy to be bought

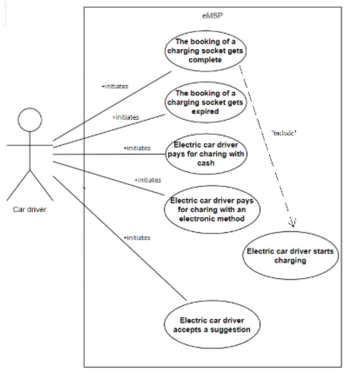
3.2 System requirements and constraints

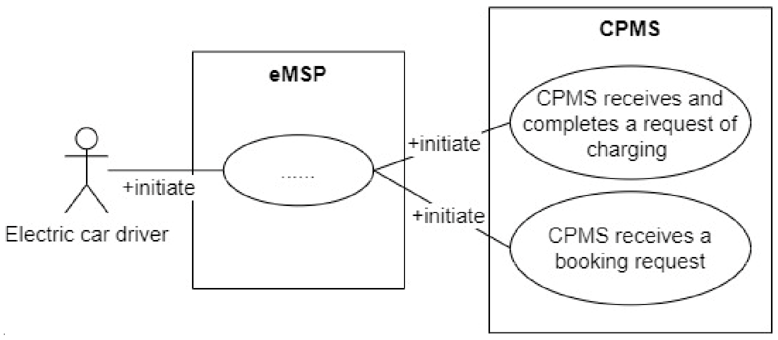
3.2.1 Use case diagrams

1. Unregistered car driver

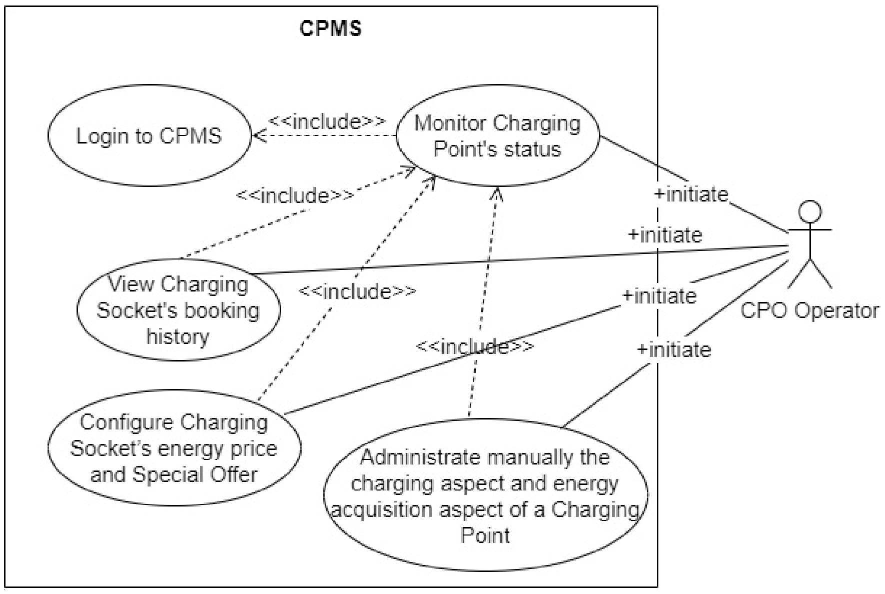


2. Car driver

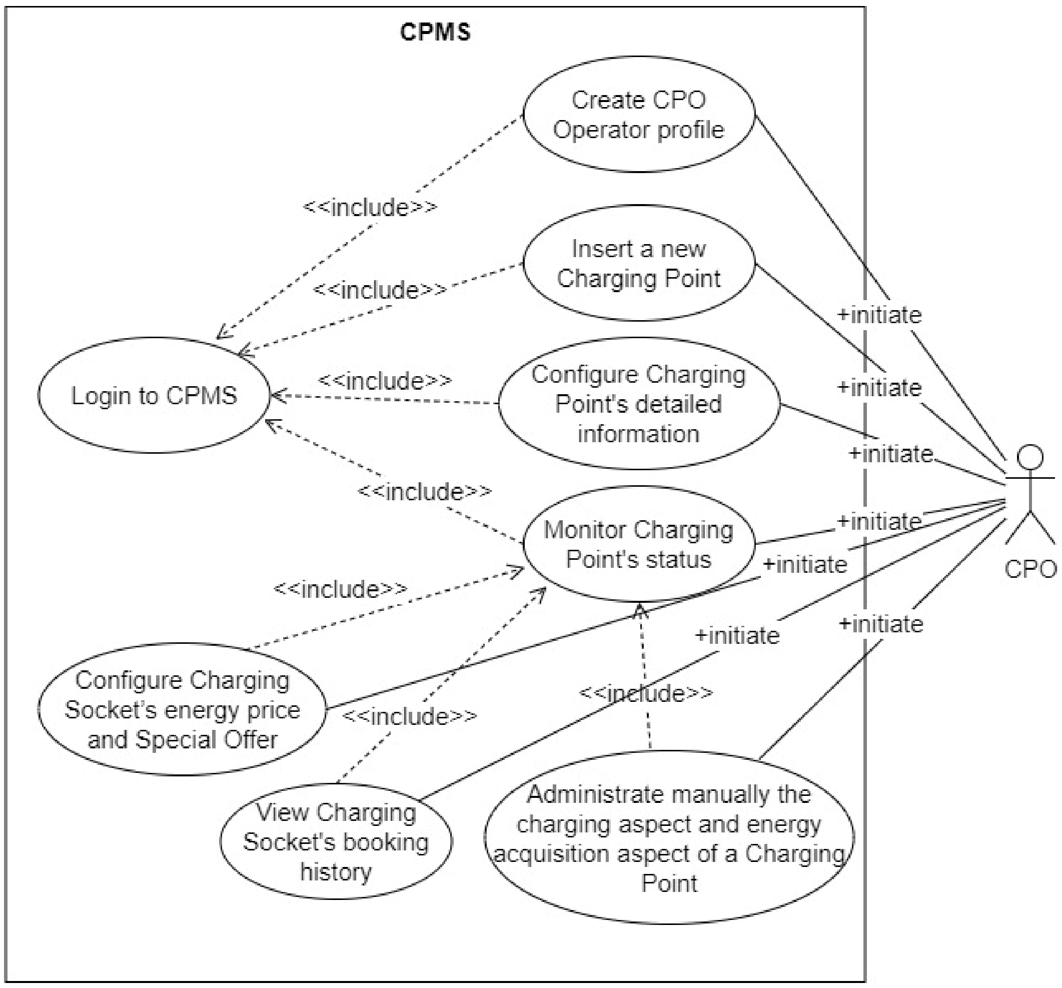

eMSP
The booking of a charging socket gets complete
The booking of a charging socket gets expired
Electric car driver pays for charing with cash
Electric car driver pays for charing with an electronic method
Electric car driver starts charging
Electric car driver accepts a suggestion
+initiates
+initiates
+initiates
+initiates
+initiates
"include"
Car driver


eMSP
CPMS
......
+initiate
+initiate
+initiate
CPMS receives and completes a request of charging
CPMS receives a booking request
Electric car driver

3. CPO Operator


CPMS
Login to CPMS
<<include>>
Monitor Charging Point's status
<<include>>
View Charging Socket's booking history
<<include>>
<<include>>
Configure Charging Socket's energy price and Special Offer
Administrate manually the charging aspect and energy acquisition aspect of a Charging Point
+initiate
+initiate
+initiate
+initiate
CPO Operator

4. CPO


CPMS
Create CPO Operator profile
Insert a new Charging Point
Login to CPMS
Configure Charging Point's detailed information
Monitor Charging Point's status
Configure Charging Socket's energy price and Special Offer
View Charging Socket's booking history
Administrate manually the charging aspect and energy acquisition aspect of a Charging Point
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
+initiate
+initiate
+initiate
+initiate
+initiate
+initiate
+initiate
CPO

3.2.2 Use cases

Use cases from eMSP system's point of view

1. **Electric car driver registration**

Actor	**Unregisterd electric car driver**
Entry conditions	The car driver doesn’t have a profile and is on the initial view of the eMSP mobile application
Event flow	1. The car driver presses the “Create a profile” button entering the dedicated section 2. The car driver enters name, surname, birth date, email address, password, debit card details, driving licence details 3. The car driver enters the car details: model, electronic identifier 4. eMSP takes up to an hour to processes the information asynchronously 5. When the information is processed the eMSP notifies the car driver about a successful registration
Exit conditions	User gets registered
Exeptions	1. Lost internet connection 2. Smatphone ran out of battery 3. Car model not found 4. Driving licence details are invalid 5. debit card details are invalid - In all cases the app notifies the driver |

2. The booking of a charging socket gets complete

Actor	**Electric car driver**
Entry conditions	The car driver intends to charge the car
Event flow	1. The car driver opens the eMSP mobile application 1. In the main menu, the driver clicks on “find a charging point” 2. The eMSP sends information request to CMSPs of the nearby charging stations and receives the respose 3. Having entered in the map the driver sees its location on the map as well as locations of the charging stations indicated with the flags for each charging socket type representing if someones of this type are available whithin the next 15 mins. 4. The driver clicks on a CPs icon and gets the description of the CP which is: number of charging sockets of each type and the their availablity time , current prices, any special offers associated to this CP 5. The driver clicks on an icon of a certain available charging socket and chooses the timeframe of booking. 6. The driver selects the payment method and clicks on the “book” button 7. The eMSP sends the booking request, info about the driver and the car to the CPMS of this specific charging station and receives the confimation, synchronized details of the charging 8. The app displays the message of a successful booking and the time counter before the charging should start 9. The driver arrives at the CP before the charging start time and plugs-in the charger into the car if it’s already free 10. The car driver opens the eMSP mobile application 11. The car driver is in the main menu, he clicks on the icon representing the booking. 12. After entering the booking section the car driver sees the details of the booking (i.e. booking timeframe, charging socket type, booked charging socket id, price of charging), the buttom “cancel booking”, the buttom “start charging”. 13. The car driver clicks on “start charging” 14. The eMSP send the start charging request to CPMS and charging starts 15. The eMSP receives confirmation from CPMS that charging started and notifies the driver 16. When charging is finished, the eMSP receives the notification from CPMS, and eMSP at its turn, notifies user 17. When the charging socket is freed the booking gets “complete” status
Exit conditions	The charging is complete
Exeptions	30 mins before the booking end time the eMSP estimates if the charging is going to be complete based on current battery level. If not and if there’s no another booking after the booking end time, the eMSP proposes to prolongate the booking for additional 15 mins If the driver plugs-off the charger before the booking gets “complete” status

3. **The booking of a charging socket gets expired**

Actor	**Electric car driver**
Entry conditions	The car driver intends to charge the car
Event flow	1. The car driver opens the eMSP mobile application 2. In the main menu, the driver clicks on "find a charging point" entering in the map 3. The eMSP sends information request to CMSPs of the nearby charging stations and receives the respose 4. The driver sees its location on the map as well as locations of the charging stations indicated with the flags for each charging socket type representing if someones of this type are available whithin the next 15 mins. 5. The driver clicks on a CPs icon and gets the description of the CP which is: number of charging sockets of each type and the their availablity time , current prices, any special offers associated to this CP 6. The driver clicks on an icon of a certain available charging socket and chooses the timeframe of booking. 7. The driver selects the payment method and clicks on the "book" button 8. The eMSP sends the booking request, info about the driver and the car to the CPMS of this specific charging station and receives the confimation, synchronized details of the chargingSP sends the booking request, info about the driver and the car to the CPMS of this specific charging station and receives the confimation, synchronized details of the charging 9. The app displays the message of a successful booking and the time counter before the charging should start 10. The driver doesn't arrive at the charging station by the time the charging should start 11. The CMSP detects the fact that the charging socket isn't connected to a car and triggers the booking cancelation sending the notification about to the eMSP 12. The booking is canceled, gets status "expired" and the driver is notified
Exit conditions	The charging booking is expired and canceled
Exeptions	

4. **Electric car driver pays for charging with cash**

Actor	**Electric car driver**
Entry conditions	The booking of a charging socket is complete and the car is charged. During the booking procedure the car driver chose “cash” as a payment method. The car driver has just disconnected the car from the charging socket
Event flow	1. The car driver opens the eMSP mobile application 2. The car driver is in the main menu, he clicks on the icon representing the booking he’s just completed 3. The car driver seeing the status of the charging “complete”, clicks on the “pay” button entering the payment menu. 4. The car driver sees the payment method which he chose during the booking procedure and buttons “confirm and pay” and “change payment method”. 5. The car driver sticks to the previously chosen method and he clicks “confirm and pay” 6. The eMSP sends the request to the CPMS of getting the identifier of the charging 7. The car driver is shown an identifier of the charging he got. 8. The car driver comes to one of the cash deposit machines mounted in the charging station. 9. He/she enters the charging identifier he saw in the eMSP app and pays for the service 10. The driver sees the “successfull payment” meassage on the screen of the cash deposit machine
Exit conditions	The user has paid for the charging
Exeptions	The car driver changed his mind and clicked “change payment method”. In this case, he/she selects the payment method and reenters the payment menu.

5. **Electric car driver pays for charging with an electronic method**

Actor	**Electric car driver**
Entry conditions	The booking of a charging socket is complete and the car is charged. During the booking procedure the car driver chose electronic payment method (card, paypal).The car driver has just disconnected the car from the charging socket
Event flow	1. The car driver opens the eMSP mobile application 2. The car driver is in the main menu, he clicks on the icon representing the booking he's just completed 3. The car driver seeing the status of the charging "complete", clicks on the "pay" button entering the payment menu. 4. The car driver sees the payment method which he chose during the booking procedure and buttons "confirm and pay" and "change payment method". 5. The car driver sticks to the previously chosen method and he clicks "confirm and pay" 6. The eMSP synchronizes the payment with the payment system 7. The payment process has been complete and the driver sees the "successful payment" message
Exit conditions	The user has paid for the charging
Exeptions	The car driver changed his mind and clicked "change payment method". In this case, he/she selects the payment method and reenters the payment menu. The payment procesure isn't complete (probably the user doesn't have enough money on his card/paypal). In this case, the car driver is notified

6. **Electric car driver starts charging**

Actor	**Electric car driver**
Entry conditions	The booking of a charging socket is accepted. The car driver got to the charging station and connected his car to the specific charging socket he/she booked.
Event flow	1. The car driver opens the eMSP mobile application 2. The car driver is in the main menu, he clicks on the icon representing the booking. 3. After entering the booking section the car driver sees the details of the booking (i.e. booking timeframe, charging socket type, booked charging socket id, price of charging), the buttom “cancel booking”, the buttom “start charging”. 4. The car driver clicks on “start charging” 5. The eMSP send the start charging request to CPMS 6. The eMSP receives the resoponse from the CPMS 7. The eMSP notifies the user that the charging has started
Exit conditions	The process of charging has started
Exeptions	The charging socket isn’t connected to the car tighlty and the CPMS doesn’t recognize the connection. The car driver receives a notification that the charging socket isn’t connected to a car. The car driver verifies that the car is connected tightly. Then the car driver clicks on “start charging” again. The process of charging starts.

7. Electric car driver accepts a suggestion

Actor	Electric car driver
Entry conditions	The car driver is registered to eMSP, eMSP can monitor the location and the car battery level at this moment, eMSP has access to the driver schedule
Event flow	1. The car driver is driving the car 2. The eMSP synchronizes the info about the nearby charging stations with the corresponding CPMSs. 3. The eMSP processes the received info with its recommendation algorithm 4. During the drive the eMSP notifies user that there's a suggestion. 5. The car driver clicks on the notificaition on the smartphone screen 6. The car driver enters the section describing the details of the suggestion which are the current car buttery level, distance to the charging station of interest, offers available by this station. Also the section displays the buttons "accept" and "decline". 7. The car driver finds the suggestion intreresting and clicks on "accept" button entering the section describing the charging station.
Exit conditions	The car driver considering a booking of a charging socket at the charging station
Exeptions	

Use cases from CPMS system's point of view

8. Login to CPMS

Actor	CPO Operator
Entry conditions	CPO Operator has the credentials provided by their company. In case of CPO, it has the credential provided by CPMS provider.
Event flow	CPO Operator open CPMS application. 1. CPMS display Authentication view. 2. CPO Operator insert username and password and clicks on Login button. 3. CPMS validate credential CPMS display the initial view containing the list of charging points available.
Exit conditions	CPO Operator is logged in CPMS, and visualizes the initial view
Exeptions	1. Provided username and password do not corresponds to any profile. - CPMS pop-up an error window.

9. Create CPO Operator profile

Actor	CPO
Entry conditions	CPO is logged on to CPMS and on initial view
Event flow	1. CPO clicks on “Create operator profile” button. 2. CPMS render a Form window asking CPO to insert username and password for new profile. 3. CPO fills the fields and clicks on “Confirm” button. 4. CPMS insert new Operator profile in its database, displays a success message, then return to initial view.
Exit conditions	a new Operator profile is created
Exeptions	1. Creation failed due to duplicated value of Username. - CPMS pop-up an error message, then return to initial view.

10. Insert a new Charging Point

Actor	CPO
Entry conditions	CPO is logged on to CPMS and on initial view
Event flow	1. CPO clicks on “Insert new charging point” in initial view. 2. CPMS insert a new blank charging point elment into the list at initial view.
Exit conditions	List of charging poinst updated with a new blank element
Exeptions	

11. **Configure Charging Point's detailed information**

Actor	**CPO**
Entry conditions	CPO is logged on to CPMS and on initial view
Event flow	1. CPO clicks on one of the Charging Point from list in initial view. 2. CPMS read through sensor on batteries, if present, the status (capacity, current battery level, charging status) of them in that Charging Point. 3. CPMS collect the information through sensor on charging sockets, about number of vehicles being charged, amount of power absorbed, and time left to the end of the charge for each vehicles connected to sockets in that Charging Point. 4. CPMS render informations collected at previous 2 points, the source of energy currently used for charging, the mode of energy source controlling, and information of Charging Points itselft in User Interface. 5. CPO clicks the Edit button. 6. CPMS makes editable the fields regarding Charging Point (Address, Battery capacity, Store Energy In Battery Condition, Use Energy In Battery Condition, Price and Special Offer Strategy). 7. CPO edits the fields and confirm the modifications. 8. CPMS saves the modifications and refreshes the page. 9. CPO clicks on “Insert new charging socket” in Charging Point detail page. 10. CPMS will add to list a blank charging socket element. 11. CPO clicks on Edit button next to a charging socket element. 12. CPMS makes fields of that charging socket editable. 13. CPO edits the fields and confirm the modifications. 14. CPMS saves the modifications and refreshes the page.
Exit conditions	View of Detail information of a Charging Point is updated
Exeptions	1. CPO insert invalid value in some field, e.g. negative value for battery capacity. Invalid Condition string. - CPMS raise a Invalid input error pop-up, and roll back changes made.

12. **Monitor Charging Point's status**

Actor	**CPO Operator**
Entry conditions	CPO Operator is logged on to CPMS and on initial view
Event flow	1. CPO Operator select one of the Charging Points from the list in initial view. 2. CPMS read through sensor on batteries, if present, the status (capacity, current battery level, charging status) of them in that Charging Point. 3. CPMS collect the information through sensor on charging sockets, about number of vehicles being charged, amount of power absorbed, and time left to the end of the charge for each vehicles connected to sockets in that Charging Point. 4. CPMS render informations collected at previous 2 points, the source of energy currently used for charging, the mode of energy source controlling, and information of Charging Points itselft to CPO Operator through User Interface.
Exit conditions	Detailed informations about a Charging Point are shown.
Exeptions	1. CPMS failed at reading sensors data. - CPMS notifies the failure, and render other informations it seccessfully collected.

13. **View Charging Socket's booking history**

Actor	**CPO Operator**
Entry conditions	CPO Operator is logged on to CPMS and on view displaying Detailed informations about a Charging Point
Event flow	1. CPO Operator select one of the Charging Sockets from the list present at view. 2. CPMS retrieves the all booking information associated to that Charging socket and display them as a list orderd by date, starting from most recent. 3. CPO Operator can scroll the list.
Exit conditions	History booking informations about a Charging socket is shown.
Exeptions	

14. **Administrate manually the charging aspect and energy acquisition aspect of a Charging Point**

Actor	**CPO Operator**
Entry conditions	CPO Operator is logged on to CPMS and on the view showing the detail of a Charging Point
Event flow	1. CPO Operator clicks on mode of energy source controlling in order to switch it to Manual mode. 4. CPMS stops verifying the conditions basing on which it decides the source of energy used for charging, without altering the previous decisions. 5. CPO Operator clicks on source of energy to view the possibilities. 6. CPMS collect the current price of energy from available DSOs, and display them in a new view, including “Battery” element if batteries available. 7. CPO Operator chooses one or many of the option from list and confirm the choice. 8. CPMS switch the source of energy to specified option(s) and goes back to view of details about Charging Point refreshing the page and collecting updated data. 9. CPO Operator clicks on battery charging status field. 10. CPMS switch the charging status field to Charging or Not Charging, and start or stops charging batteries accordingly.
Exit conditions	Updated detailed informations about a Charging Point are shown.
Exeptions	1. CPMS failed at switching energy source. - CPMS notifies the failure, and goes back to view of details about Charging Point without altering any configuration.

15. **CPMS receives a booking request**

Actor	**Electric car driver**
Entry conditions	
Event flow	1. Electric car driver clicks on the “Book” button on his eMSP application. 2. eMSP send the booking request to CPMS with necessary information. 3. CPMS save the corresponding informations (booking - car - eMSP), and send back a success message.
Exit conditions	new Booking, Car and eMSP are created and/or associated
Exeptions	1. When booking request arrives, no socket is free in specified time slot anymore. - send back an error message.

16. **CPMS receives and completes a request of charging**

Actor	**Electric car driver**
Entry conditions	Relative booking information is present in CPMS and it has not expired yet.
Event flow	1. Electric car driver arrives to charging point, and plugs in the charging socket to his car. 2. Charging socket connects with car (physically or via Bluetooth) retrieving its electronic identifier and transmit it to CPMS. 3. Electric car driver initiate the “Start charging” request on his eMSP software, which transmit the request to CPMS. 4. CPMS receive the request, verify whether charging socket is actually connected to a car, whether there is a booking associated to such socket, and whether the electronic identifier revealed by socket matches with one registered in booking. 5. CPMS checkes energy source controlling mode and configuration of that charging point, according to which decide where it will extract energy from. 6. CPMS gives the greenlight to start the charging, and send back to eMSP a Charging started message. 7. When charging socket reveals that the car connected is fully charged, or the booking session ended, CPMS records the amount of energy consumed, stops charging and send a Charging completed message to eMSP.
Exit conditions	Relative booking’s status changed to “Completed”
Exceptions	1. Charging socket is not connected to any car. 2. No booking on that socket in that moment. 3. Car connected to socket does not match with car registered in booking. - CPMS sends back an error message. 4. Electric car driver plugs off the socket before charge process completes 5. Electric car driver clicks on “Stop charging” button on eMSP - CPMS will change booking status to “Stopped” and send a error message to eMSP waiting for further indication.

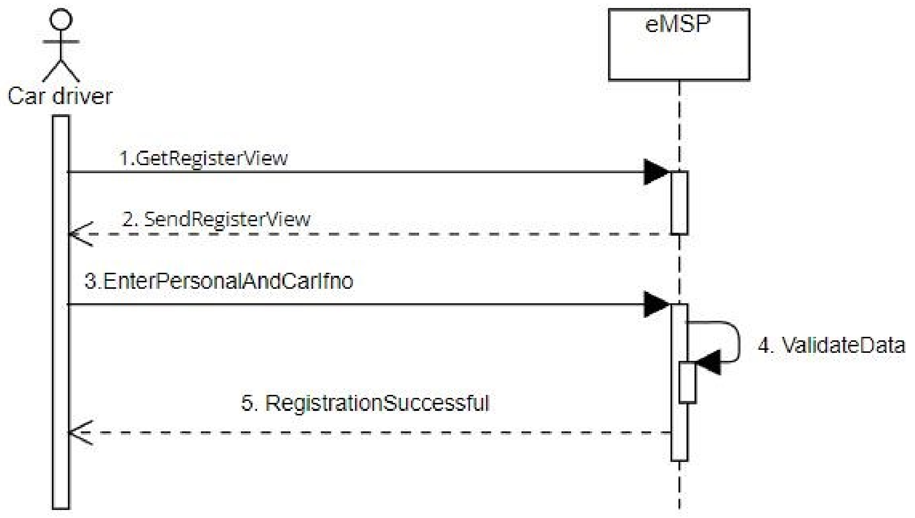
17. **Configure charging socket’s energy price and Special Offer**

Actor	**CPO Operator**
Entry conditions	CPO Operator is logged on to CPMS and on view displaying Detailed information about a Charging Point
Event flow	1. CPO Operator switch Charging Price Strategy Mode to Manual. 2. CPO Operator select one of the Charging Sockets from the list present at view.

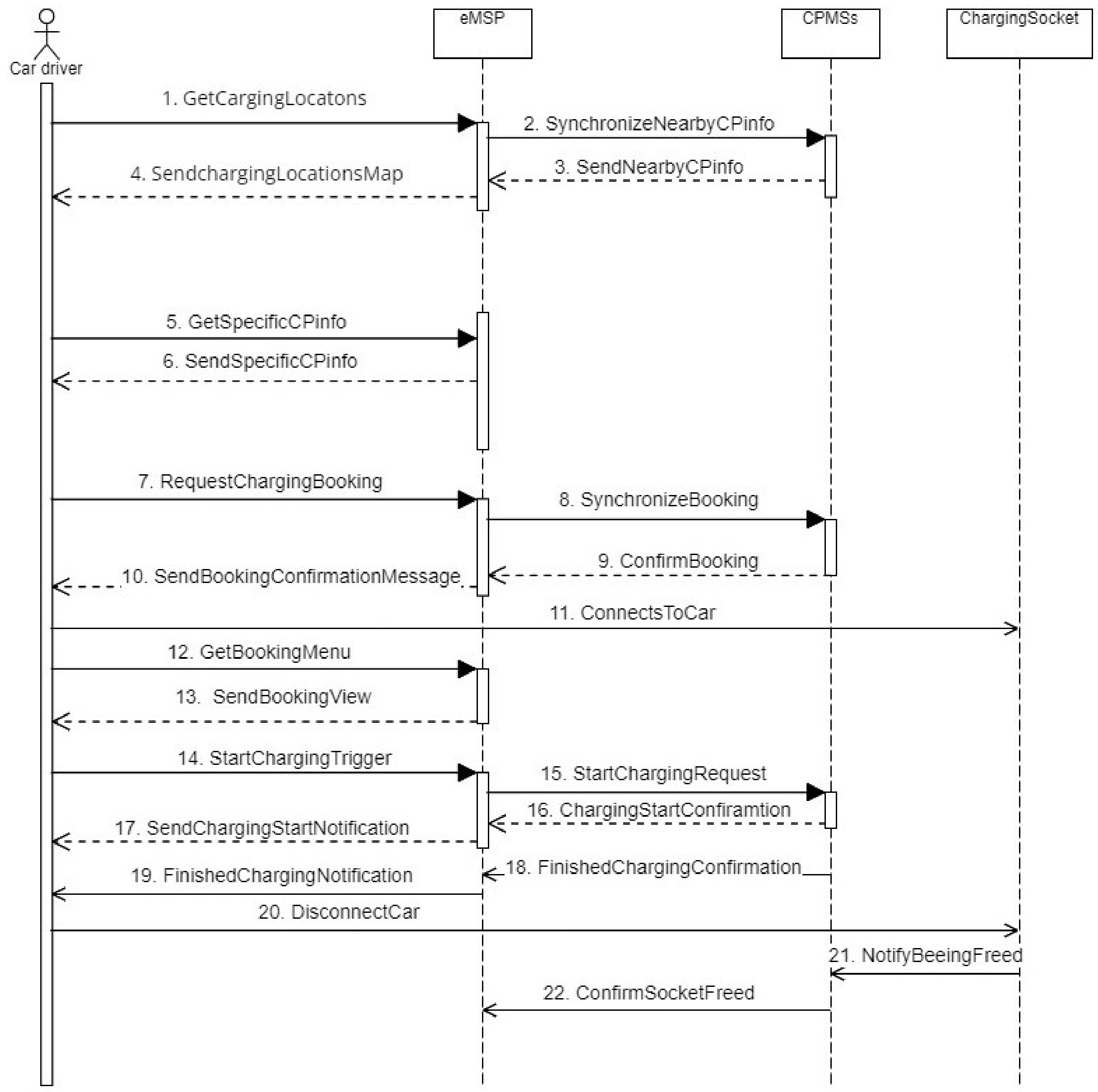
	3. CPMS retrieves all booking information associated to that Charging socket and display them as a list orderd by date, starting from most recent, current price of charging, and a button that allow user to set the price of charging and to insert Special Offer related to current socket manually. 4. CPO Operator click on button. 5. CPMS display a window with 4 fields (price of charging, discount amount, start date, end date) 6. CPO Operator fill the field(s) and confirm. 7. CPMS saves the changes made and refresh the page.
Exit conditions	New price for a Charging socket is manually set.
Exeptions	

3.2.3 Sequence diagrams

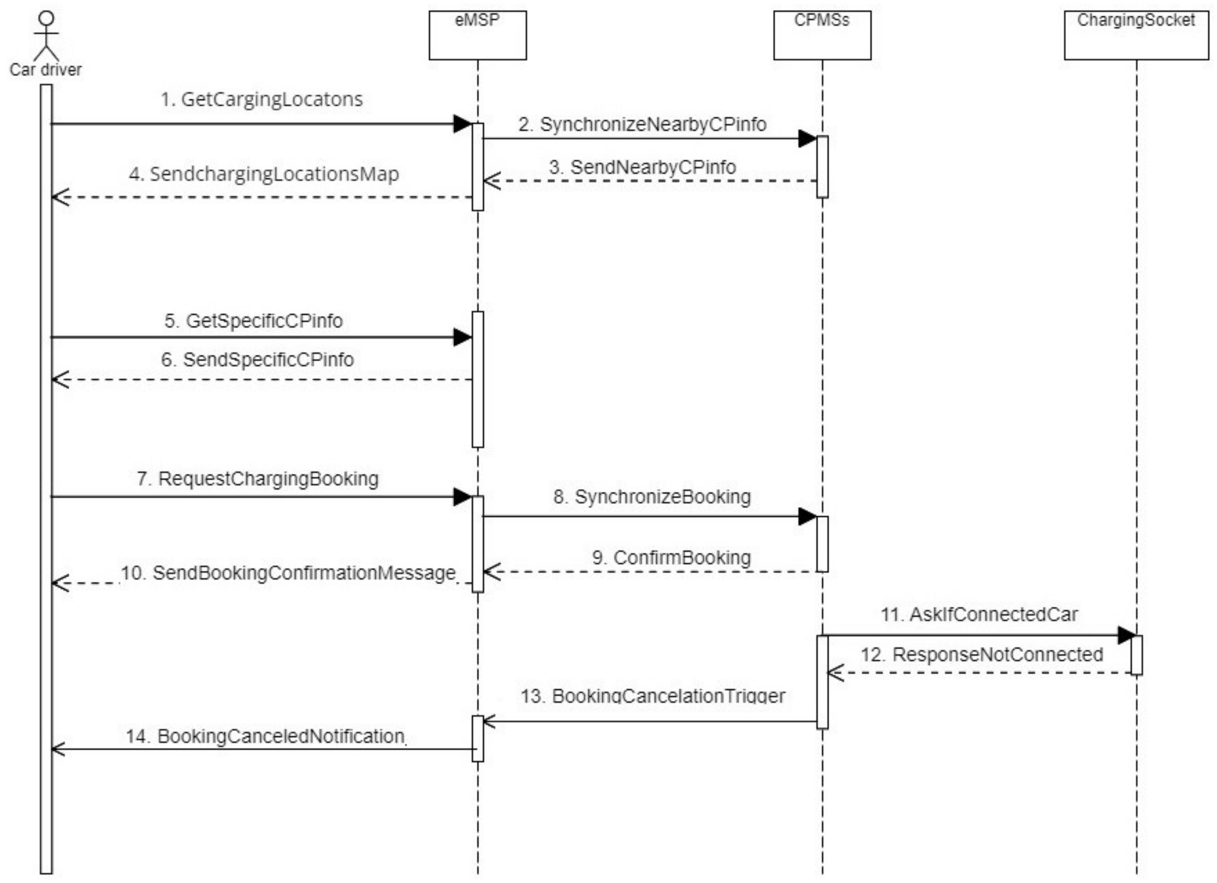
Electric car driver registration



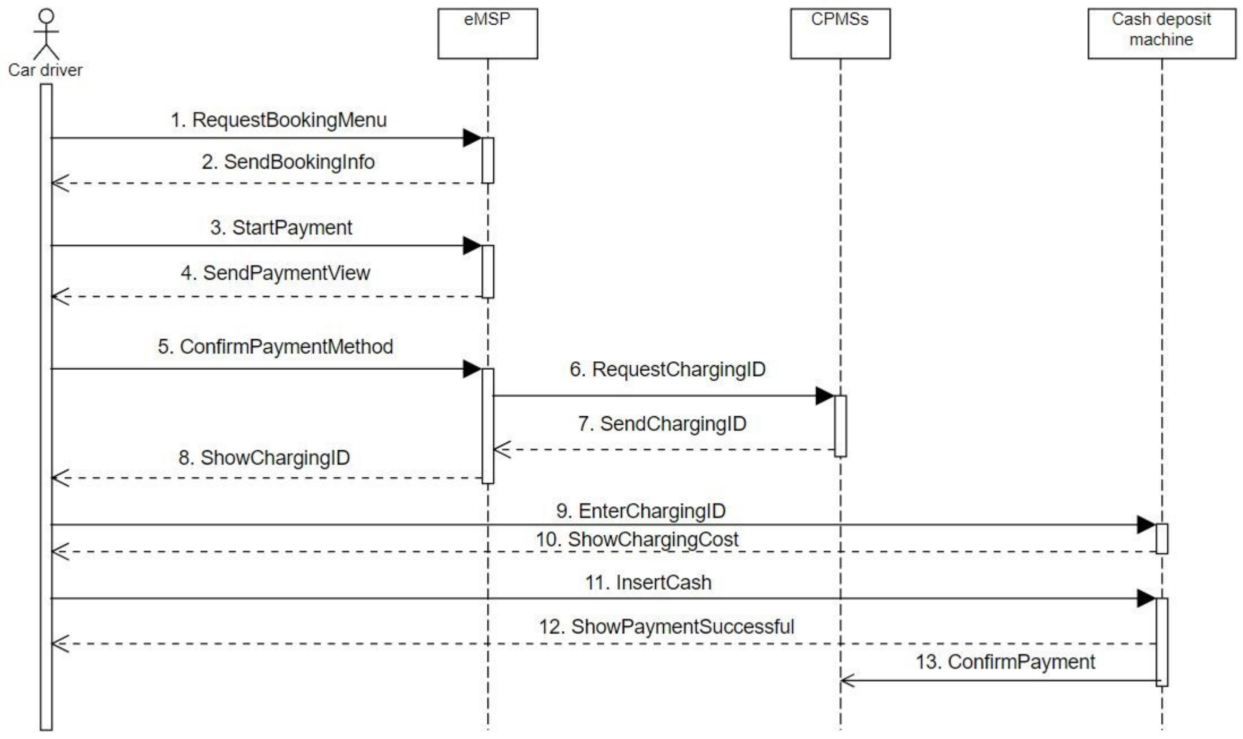
The booking of a charging socket gets complete



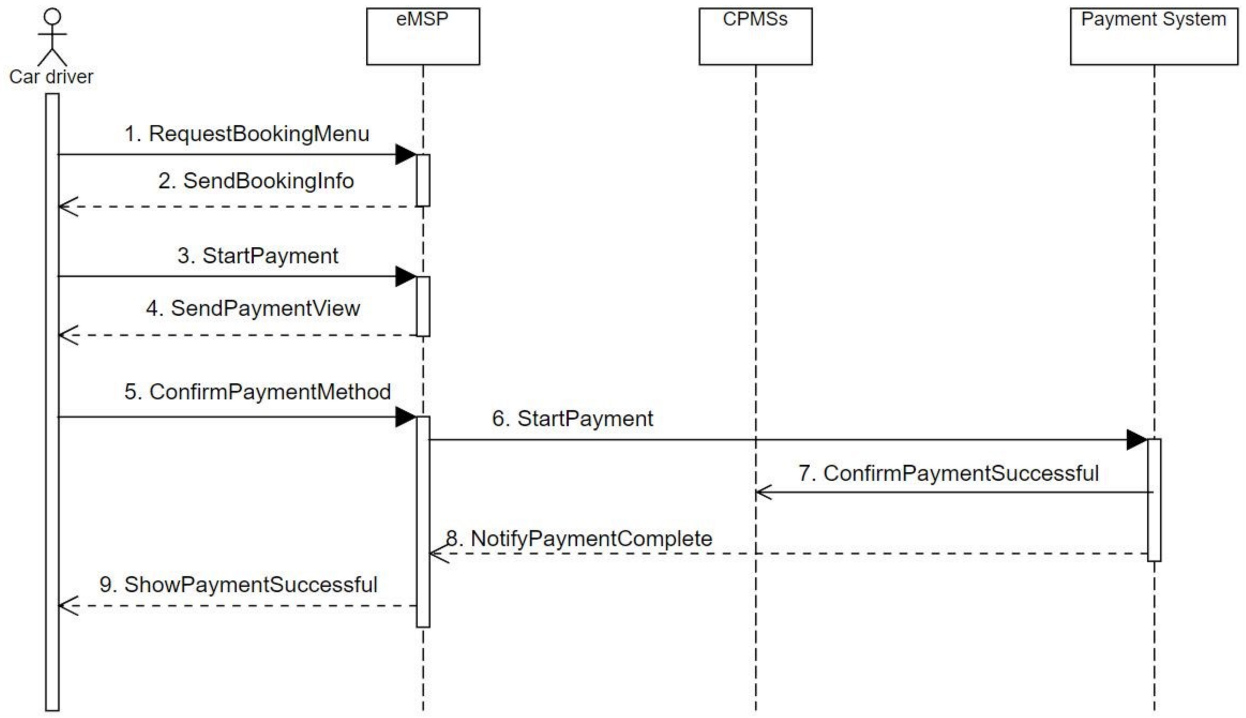
The booking of a charging socket gets expired



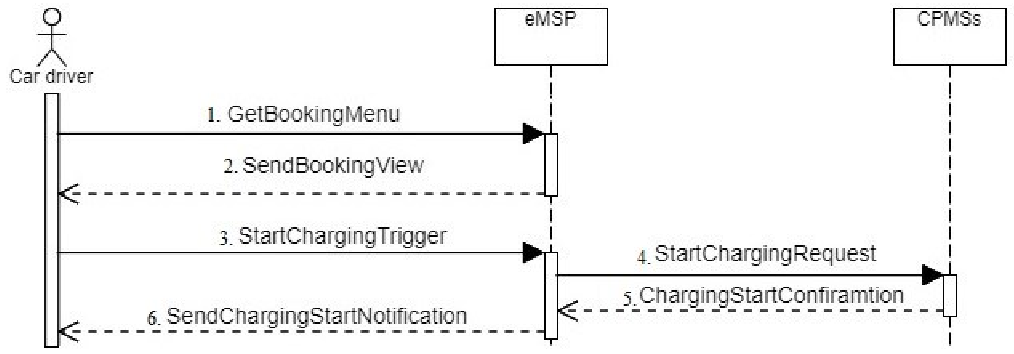
Electric car driver pays for charging with cash



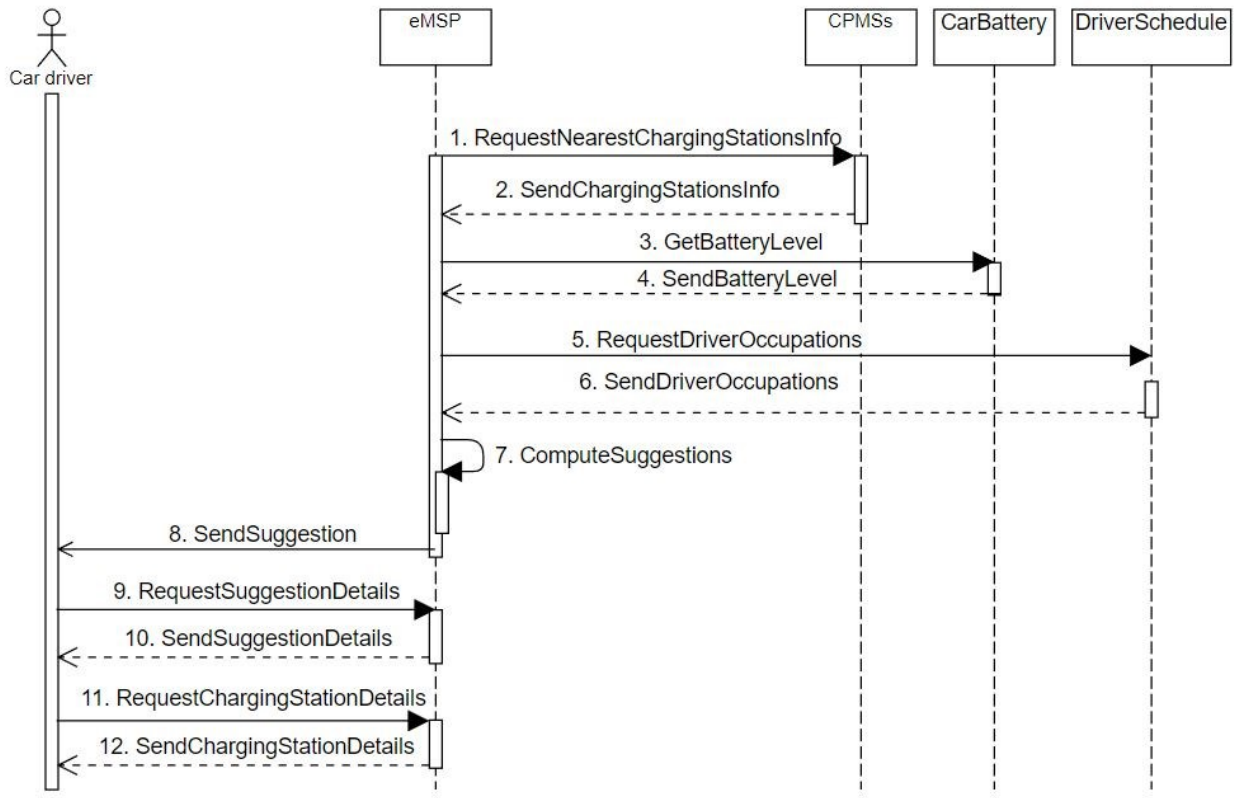
Electric car driver pays for charging with an electronic method



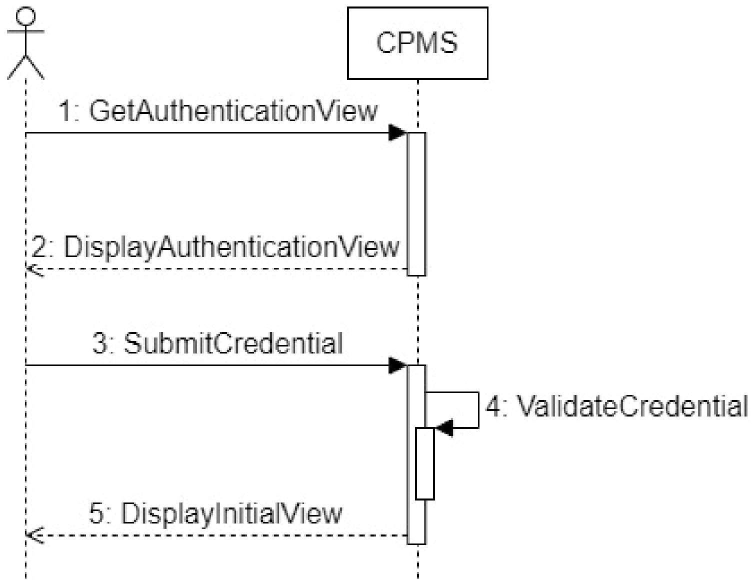
Electric car driver starts charging



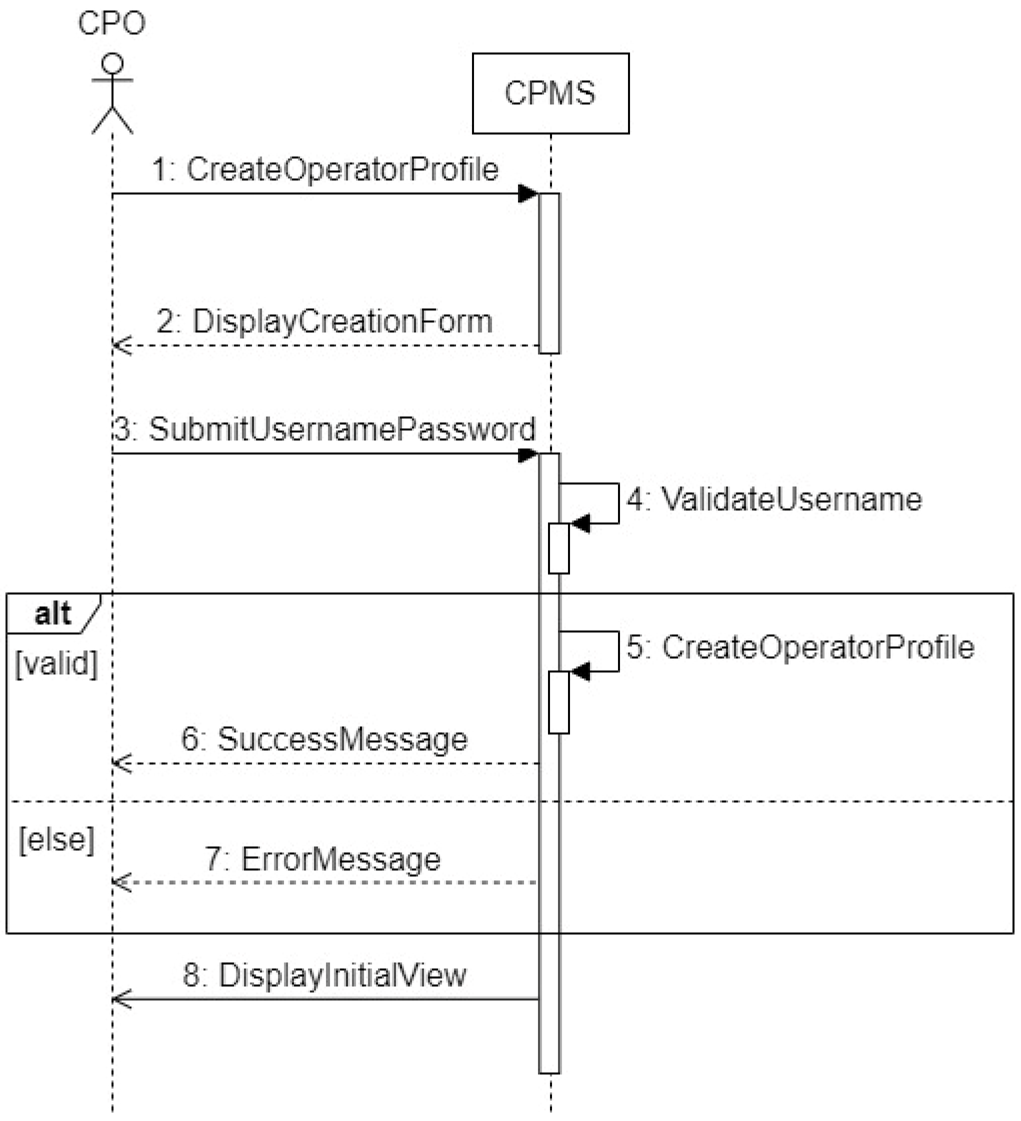
Electric car driver accepts a suggestion


Car driver
eMSP
CPMSs
CarBattery
DriverSchedule
1. RequestNearestChargingStationsInfo
2. SendChargingStationsInfo
3. GetBatteryLevel
4. SendBatteryLevel
5. RequestDriverOccupations
6. SendDriverOccupations
7. ComputeSuggestions
8. SendSuggestion
9. RequestSuggestionDetails
10. SendSuggestionDetails
11. RequestChargingStationDetails
12. SendChargingStationDetails

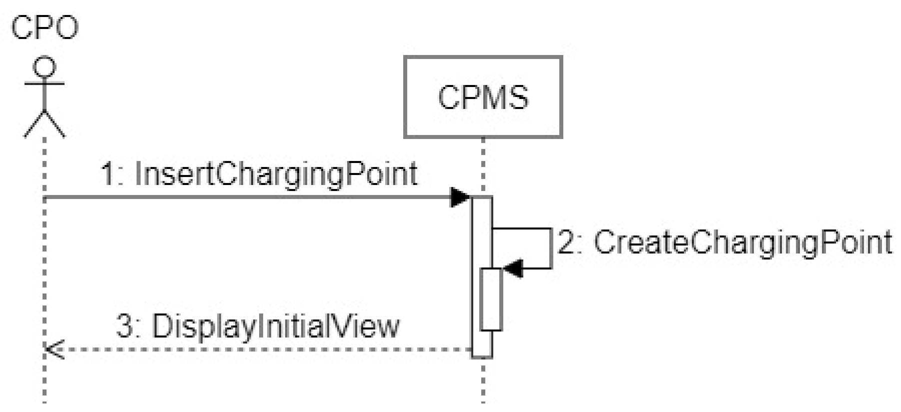
Login to CPMS



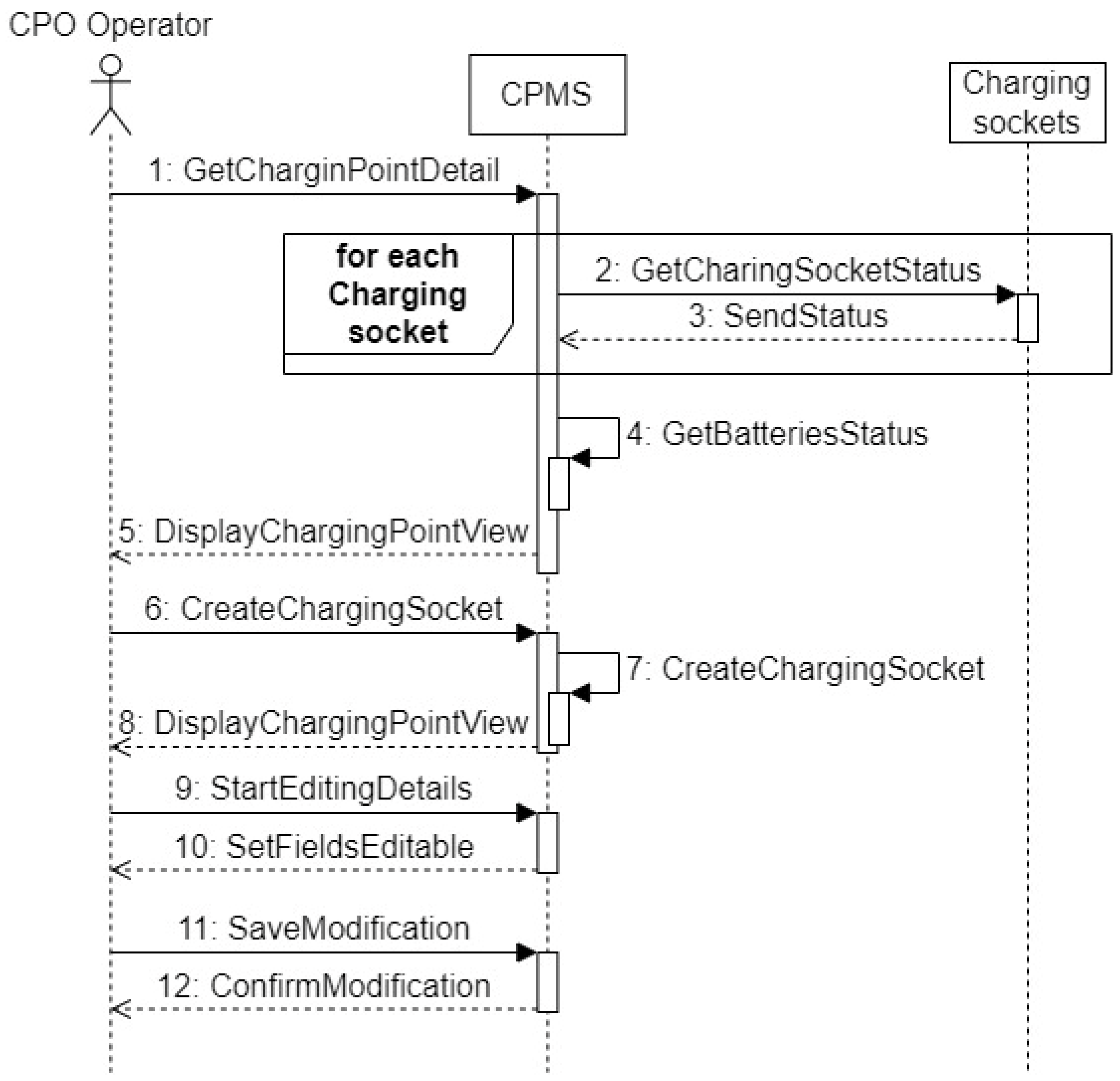
Create CPO Operator profile



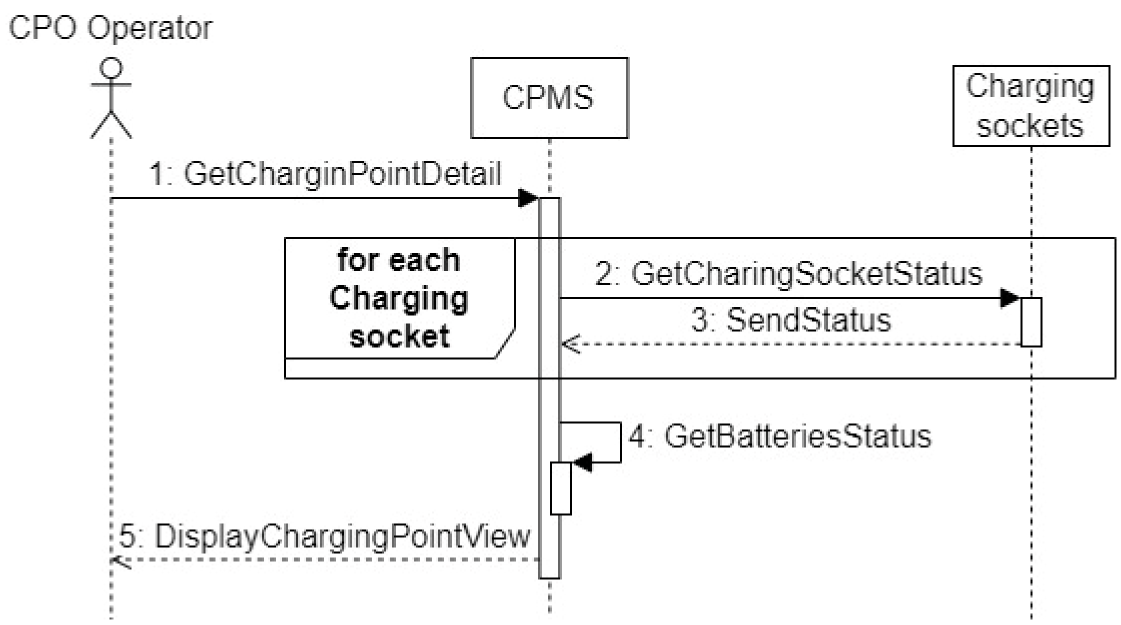
Insert a new Charging Point



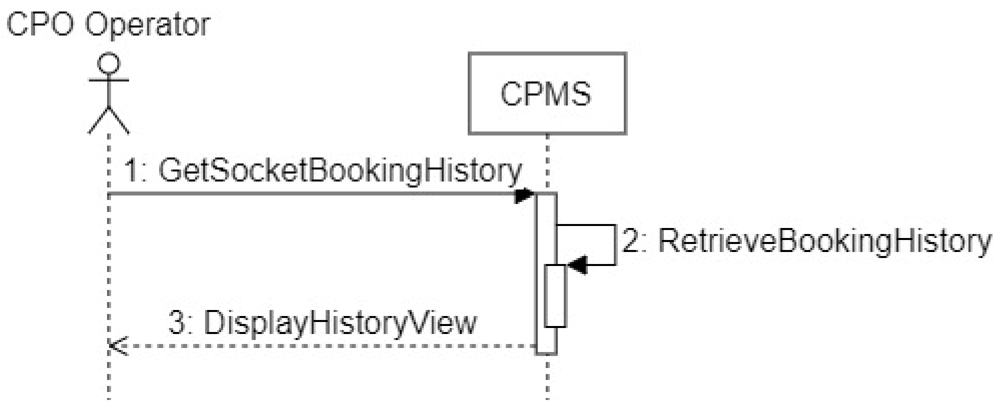
Configure Charging Point's detailed information



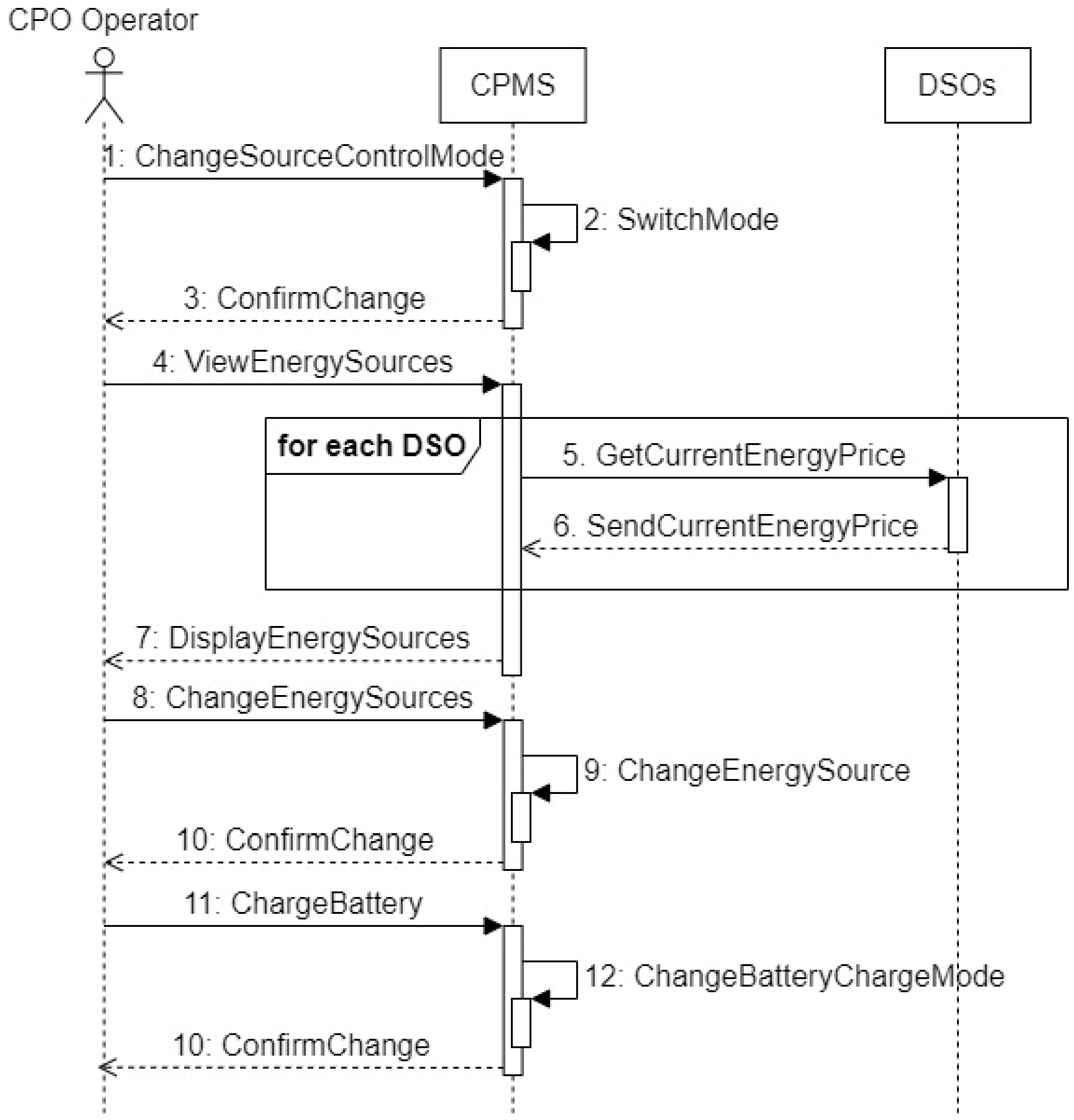
Monitor Charging Point's status



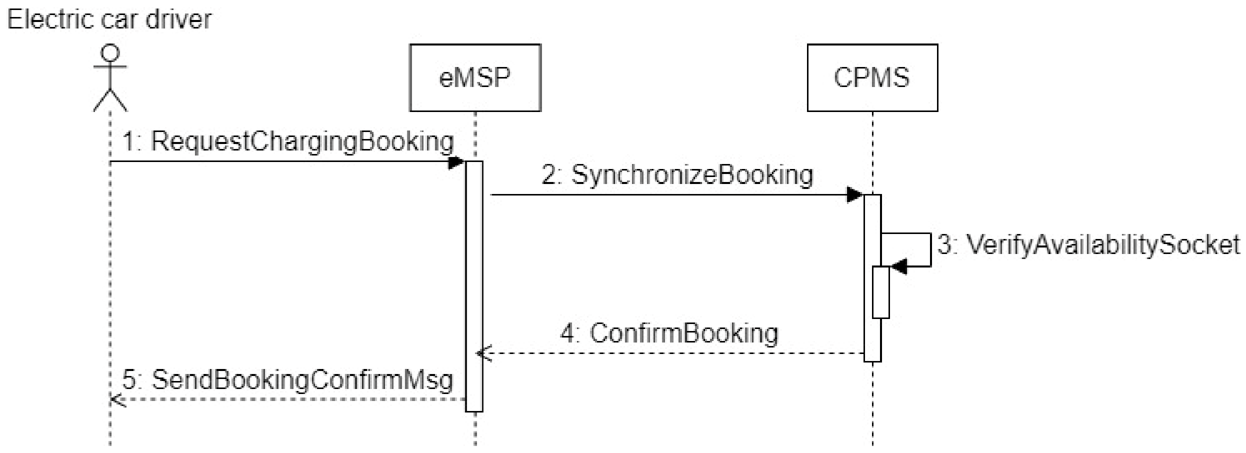
View Charging Socket's booking history



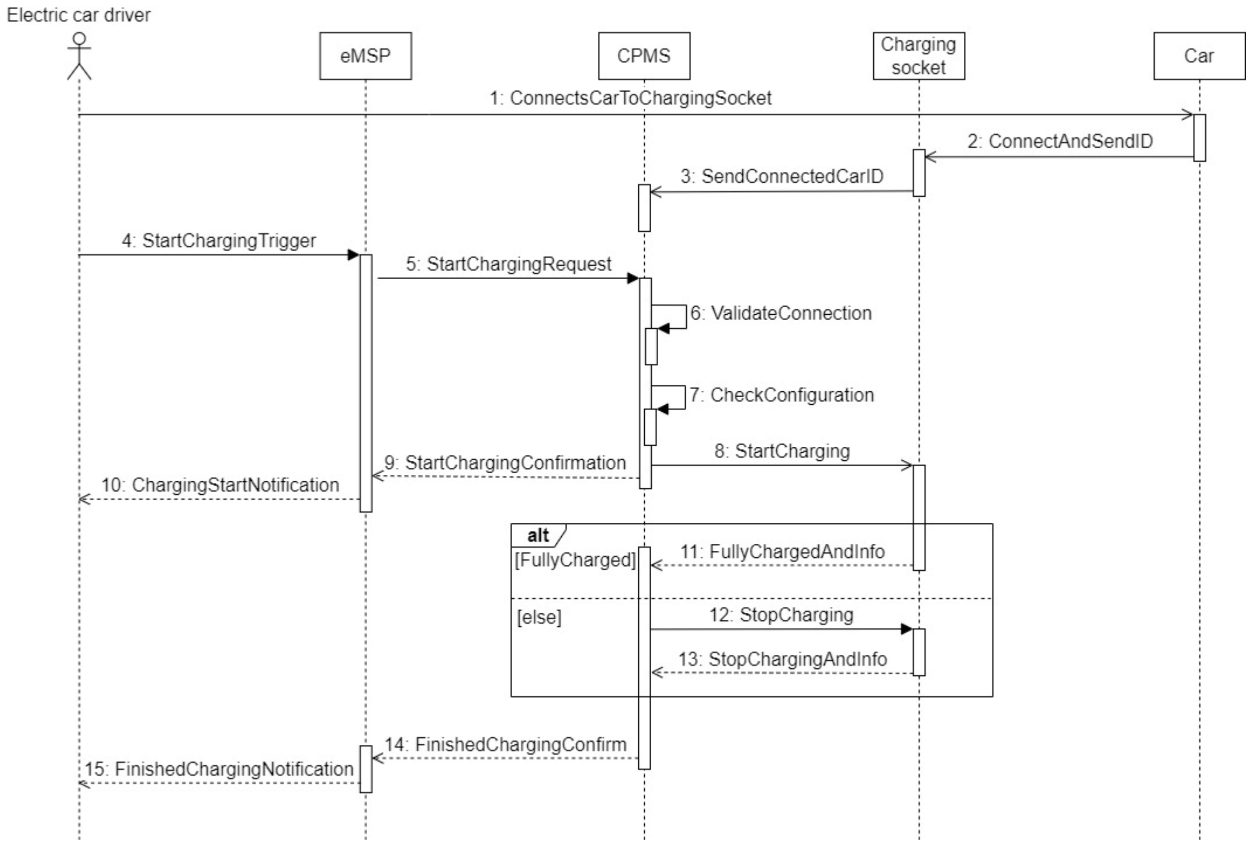
Administrate manually the charging aspect and energy acquisition aspect of a Charging Point



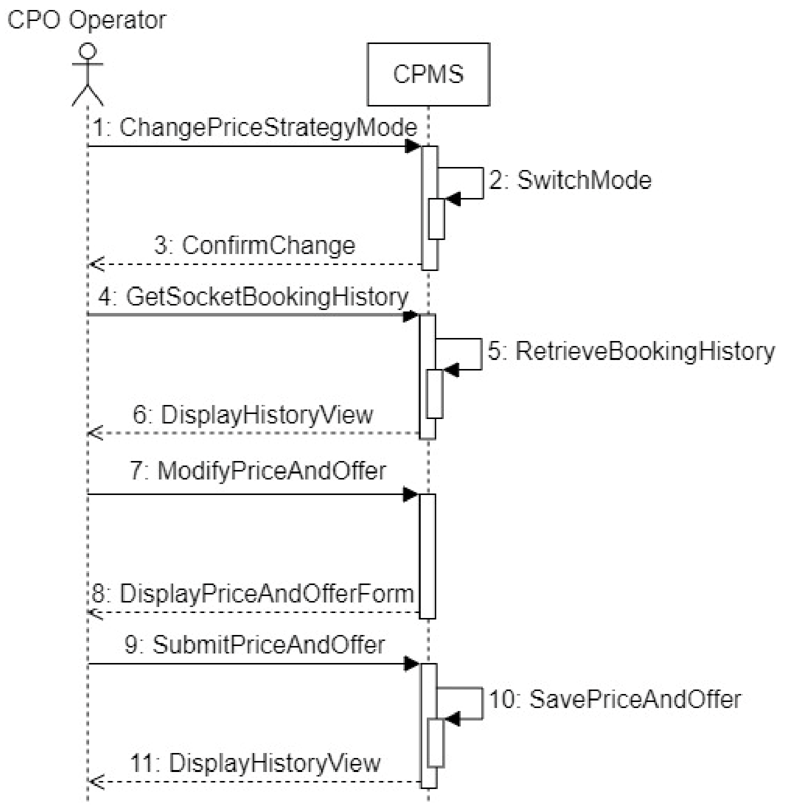
CPMS receives a booking request



CPMS receives and completes a request of charging



Configure charging socket's energy price and Special Offer



3.2.4 Mapping of Use Cases on Requirements

Login to CPMS	R1(CPMS)
Create CPO Operator profile	R2(CPMS)
Insert a new Charging Point	R3(CPMS)
Configure Charging Point's detailed information	R4(CPMS), R5(CPMS), R6(CPMS), R7(CPMS), R8(CPMS)
Monitor Charging Point's status	R9(CPMS), R10(CPMS)
View Charging Socket's booking history	R11(CPMS)
Administrate manually the charging aspect and energy acquisition aspect of a Charging Point	R12(CPMS), R13(CPMS), R14(CPMS), R15(CPMS), R16(CPMS)
CPMS receives a booking request	R17(CPMS)
CPMS receives and completes a request of charging	R18(CPMS)
The booking of a charging socket gets complete	R2(eMSP),R3(eMSP),R4(eMSP),R6(eMSP),R7(eMSP),R8(eMSP),R12(eMSP),R14(eMSP)
The booking of a charging socket gets expired	R2(eMSP),R3(eMSP),R6(eMSP),R7(eMSP),R14(eMSP)
Electric car driver pays for charging with cash	R9(eMSP)
Electric car driver pays for charging with an electronic method	R9(eMSP)
Electric car driver starts charging	R4(eMSP),R5(eMSP),R6(eMSP),R7(eMSP),R12(eMSP)
Electric car driver accepts a suggestion	R10(eMSP),R14(eMSP)

3.2.5 Functional Requirements

Requirement	**Description**
R1(eMSP)	The eMSP subsystem shall allow the User to see the estimated time before a charging socket of a certain type at a certain CP is freed
R2(eMSP)	The eMSP subsystem shall allow the user to book a charging socket of a certain type at a certain charging station for a certain timeframe maximum 15 min before the charging should start
R3(eMSP)	The eMSP subsystem shall allow the user to choose a payment method (i.e. cash, card, paypal) during the booking procedure

Requirement	Description
R4(eMSP)	The eMSP subsystem shall allow the user to trigger the start of a charging process when the car is connected to a charging socket
R5(eMSP)	The eMSP subsystem shall not allow the user to trigger the start of a charging process when the car is not connected to a charging socket
R6(eMSP)	The eMSP subsystem shall allow user to know the locactions of the charging stations within the range of 15 min reachability according to the car navigation system
R7(eMSP)	The eMSP subsystem shall allow user to know about the costs and availability of charging at charging stations within the range of 15 min reachability according to the car navigation system
R8(eMSP)	The eMSP subsystem shall allow user to know about the special offers of charging at charging stations within the range of 15 min reachability according to the car navigation system
R9(eMSP)	The eMSP subsystem shall notify user when the charging is finished
R10(eMSP)	The eMSP subsystem shall have the payment interfaces for user so that he can manage payment method details, receive receipts of payments
R11(eMSP)	The eMSP subsystem shall analyse the battery charge level, occupations of the user according to the calendar, special offers by the CPOs and avalability of charging sockets in order to make a suggestion to the user of getting a charge at a certain charging staiton
R12(eMSP)	The eMSP subsystem shall allow user to create an account and use its functionalities
R1(CPMS)	The system must allow registered Operator and CPO to login
R2(CPMS)	The system must allow CPO company to create Operator profile for its employee
R3(CPMS)	The system must allow CPO profile to create Charging Point
R4(CPMS)	The system must allow CPO profile to update a Charging Point's information
R5(CPMS)	The system must allow CPO profile to create Charging socket
R6(CPMS)	The system must allow CPO profile to update a Charging socket's information
R7(CPMS)	The system shall allow CPO profile to configure the conditions according to which CPMS will automatically change the source of energy used for charging in a CP between DSOs and batteries if present
R8(CPMS)	The system shall allow CPO profile to configure the conditions according to which the batteries in a CP will get charged, if present
R9(CPMS)	The CPMS shall display the information on available batteries present in a Charging Point to its user

Requirement	Description
R10(CPMS)	The CPMS shall display the information about number of vehicles being charged in a CP, amount of power absorbed, and time left to the end of the charge for each vehicle to its user
R11(CPMS)	The CPMS shall display the information about booking relative to a specific charging socket to its user
R12(CPMS)	The CPMS should get the current price of energy from available DSOs
R13(CPMS)	The system shall allow Operator profile to change the Energy source controlling Mode of charging point
R14(CPMS)	The CPMS shall display the information provided by DSOs to its user
R15(CPMS)	The system shall allow Operator profile to set the energy providers from where acquire or extract energy
R16(CPMS)	The system shall allow Operator profile to configure whether store energy in a charging point's batteries
R17(CPMS)	The system shall allow external eMSP to book a charging socket for a time period.
R18(CPMS)	The system shall allow external eMSP to start and stop a charging process on condition satisfied

3.2.6 non-Functional Requirements

R17(eMSP)	The system should penalize the driver if the car isn't disconected from the charging socket at most 15 mins after the booking end time

3.2.7 Constraints

R13(eMSP)	The eMSP subsystem must be informed about moment when the car is connected to a charging socket by the CPMS subsystem through an API
R14(eMSP)	The eMSP subsystem must be informed about moment when the car is disconnected from a charging socket by the CPMS subsystem through an API
R15(eMSP)	The eMSP subsystem must be informed about moment when the charging of the corresponding car is finished through an API
R16(eMSP)	The eMSP should be able to interact with multiple CPMS

3.2.8 Mapping on Goals

Goals	Domain assumptions	Requirements
G1(eMSP)	D6, D7, D8	R6(eMSP),R8(eMSP), R9(eMSP),R16(eMSP)
G2(eMSP)	D2, D4, D5, D6, D7, D8	R1(eMSP), R11(eMSP),R16(eMSP)

Goals	Domain assumptions	Requirements
G3(eMSP)	D2, D6,D7, D8	R1(eMSP), R2(eMSP), R3(eMSP), R6(eMSP), R16(eMSP), R14(eMSP), R15(eMSP)
G4(eMSP)	D2, D3, D5, D8, D9, D10, D11, D12, D13	R3(eMSP), R4(eMSP), R5(eMSP), R9(eMSP), R10(eMSP), R3(CPMS), R12(eMSP), R14(eMSP), R15(eMSP), R17(eMSP)
G5(eMSP)	D1, D2, D3	R12(eMSP)
G1(CPMS)	D14, D15, D18, D19	R1(CPMS), R2(CPMS), R3(CPMS), R4(CPMS), R12(CPMS), R13(CPMS), R14(CPMS), R15(CPMS)
G2(CPMS)	D14, D15, D16	R1(CPMS), R2(CPMS), R3(CPMS), R4(CPMS), R8(CPMS), R9(CPMS), R16(CPMS)
G3(CPMS)	D14, D15, D16, D18, D19	R1(CPMS), R2(CPMS), R3(CPMS), R4(CPMS), R7(CPMS), R8(CPMS), R9(CPMS), R13(CPMS), R15(CPMS)
G4(CPMS)	D17, D20, D21, D22	R17(CPMS), R18(CPMS)
G5(CPMS)	D14, D15, D16, D17	R1(CPMS), R2(CPMS), R3(CPMS), R4(CPMS), R9(CPMS), R10(CPMS), R11(CPMS)
G6(CPMS)	D16, D17, D18, D19, D20, D21, D22	R17(CPMS), R18(CPMS)

3.3 Performance Requirements

eMSP must be as simple as possible, avoid any hard calculations. All these operations should be done in some external server operated by us. We consider that the client might have not the fast internet connection while being on the road. That is why the exchange of information between eMSP and external sources (Except for the data coming from the car directly) should be minimized. Coming to *CPMS*, it obviously has higher requirements. To be more exact, it should handle asynchronously the information coming from at least 1 000 000 clients and 100 000 charging sockets (Assuming 10 000 CPs with 10 charging sockets in each)

3.4 Design Constraints

3.4.1 Standards Compliance

- All user data shall be treated in compliance with GDPR.
- The *eMSP* should runs fully on all widely used smartphones.
- *CPMS* should be accessible both from desktop and mobile devices. The conventions regarding APIs should be followed.

3.4.2 Hardware Limitations

eMSP:

Smartphone should have an access to the internet and to the car through bluetooth.

CPMS:

The devices must have access to the internet.

3.5 Software System Attributes

3.5.1 Reliability

Considering the importance of energy in our world, *eMSP* and *CPMS* must have high reliavility level. Several servers must be used in order to be able to do the maintanance works on not loaded ones. Moreover, these works should be done during the nights, when the car usage is the lowest.

3.5.2 Availability

The availability of the server is crucial, but some servers will have to be turned off due to many reasons (Maintanance works, Force majeure). As mentioned before, in order to minimize the impact of such situations, several servers should run our systems with availability to delegate its workload to other servers.

3.5.3 Security

It is obvious that we are responsible for keeping such private information of ordinary users, such as passwords, statistics of vehicle usage, the bills of a client (His spendings on energy). All this information should be stored in a crypted way. When it comes to CPOs, the whole solution (Means both frontend and backend) are given to them. From our side it is important to use stable encryption ways and provide CPOs with regular security updates.

3.5.4 Maintanability

Both *eMSP* and *CPMS* should be easy to maintain and update in future. To achieve this, there will be need in following the well-known patterns in Software Development. In addition, there must be a well-written documentation about these two software. There should be no hesitation in adding the comments directly into the source code.

3.5.5 Portability

eMSP should work on any modern smartphone, regardless of the screen size. The same client should be anle to access his/her account on both Android and Ios.

CPMS should be able to run on either Windows or Linux. CPO should be able to gain access to its account on any computer using previously mentioned Operating Systems. The interface of this application should be adapted to 16:9 resolution.

4. Formal analysis

In this section the analysis of the system using Alloy is represented. The purpose is to show that the system is satisfiable. The whole project was divided into 2 parts: eMSP and CPMS. Alloy for both of them is represented below. The Alloy was written based on the class diagrams illustrated before.

eMSP:

```
//Simple signatures

sig Name{}

sig Address{}

sig Surname{}

sig Email{}

sig Password{}

sig ElectronicIdentifier{}

//Enums

enum BookingStatus {Canceled, Accepted, Complete}


enum PaymentMethod {DebitCard, Paypal, Cash}


//ReadyToBeFreed means that the charging process has been finished but the car is still
connected to the charging socket

enum AvailabilityStatus {Free, Occupied, ReadyToBeFreed}


enum SocketType {Slow, Fast, Rapid}

//Complex signatures

sig Date{

        days_passed: Int

}


sig Time{

        minutes_passed: Int

}

//The current time is represented as a certain amount of days and minutes passed from a certain
point
```

```
sig CurrentTime{
        date: one Date,
        time: one Time
}

sig User{
        name: one Name,
        surname: one Surname,
        email: one Email,
        password: one Password
}

sig Car{
        owner: one User,
        electronic_identifier: one ElectronicIdentifier,
        battery_capacity: one Int,
        current_battery_level: one Int
}

sig Occupation{
        user: one User,
        date: one Date,
        starting_time: one Time,
        ending_time: one Time
}

sig Booking{
        car: one Car,
        charging_socket: one ChargingSocket,
        payment_method: one PaymentMethod,
        date: one Date,
        start_time: one Time,
        end_time: one Time,
        status: one BookingStatus
}

sig Suggestion{
        car: one Car,
        charging_socket: one ChargingSocket,
        expire_time: one Time
}

sig CPO{
        name: one Name,
        address: one Address
}

sig CP{
        owner: one CPO,
        address: one Address
}

sig ChargingSocket{
        owner: one CP,
        availability_status: one AvailabilityStatus,
        estimated_charging_end_time: one Int,
        socket_type: one SocketType,
        price: one Int
}
//facts
fact AllCPOsHaveCP{
        all cpo: CPO | some cp: CP | cpo in cp.owner
}
```

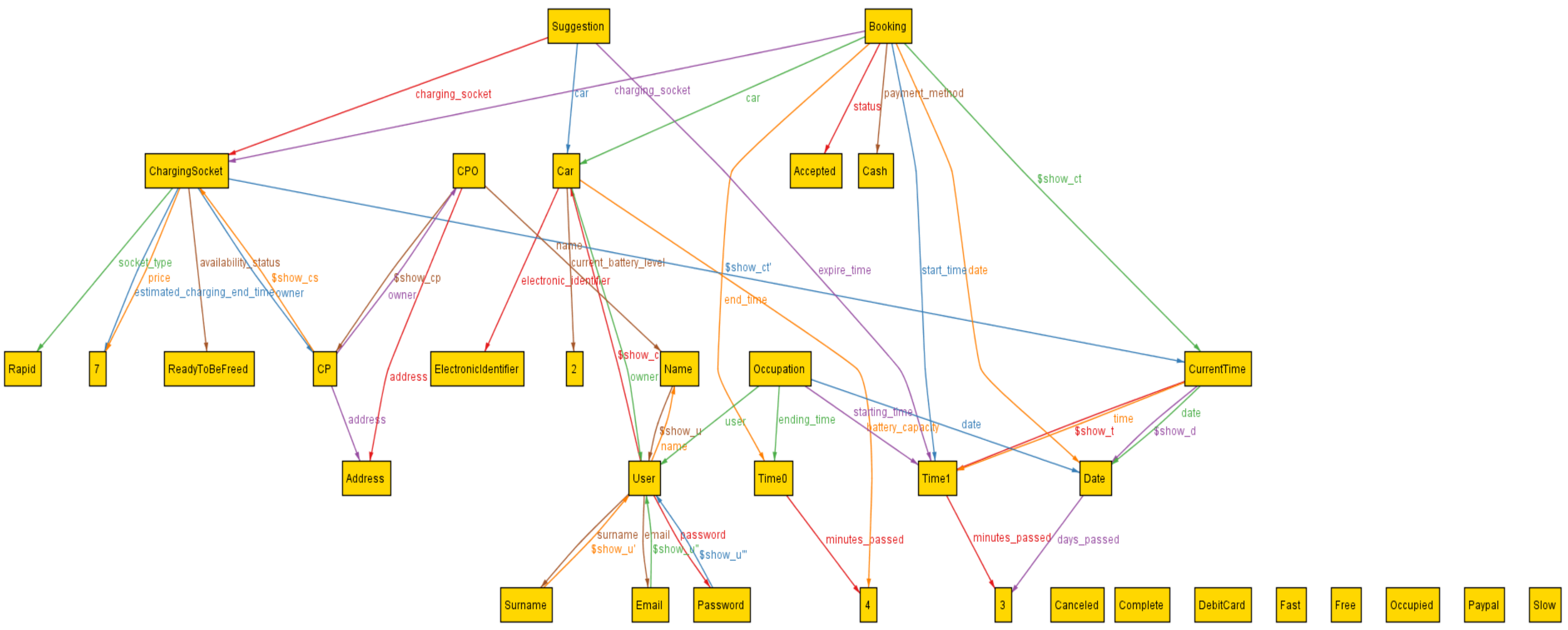
```
fact AllCPsHaveChargincSocket{
        all cp: CP | some cs: ChargingSocket | cp in cs.owner
}

fact AddressesOfCPsAreUnique{
        no disjoint cp1, cp2: CP | cp1.address = cp2.address
}

fact AddressesOfCPOsAreUnique{
        no disjoint cpo1, cpo2: CPO | cpo1.address = cpo2.address
}

fact AllUsersHaveCar{
        all u: User | some c: Car | u in c.owner
}

fact ValidValuesForBatteryInCars{
        all c: Car | (
                c.battery_capacity > 0 &&
                c.current_battery_level > 0 &&
                c.current_battery_level <= c.battery_capacity
        )
}

fact ValidValuesForDates{
        all d: Date | d.days_passed > 0
}

fact ValidValuesForTimes{
        all t: Time | t.minutes_passed > 0
}
```

```
fact ValidValuesForTimesInOccupation{
        all o: Occupation | (
                o.ending_time.minutes_passed > o.starting_time.minutes_passed
        )
}

fact ValidValuesForTimesInBooking{
        all b: Booking | (
                b.end_time.minutes_passed > b.start_time.minutes_passed
        )
}

fact UniqueEmail{
        no disjoint u1, u2: User | u1.email = u2.email
}

fact UniqueElectronicIdentifiers{
        no disjoint c1, c2: Car | c1.electronic_identifier = c2.electronic_identifier
}

fact NoBookingsAtTheSameTimeForOneCar{
        no disjoint b1, b2: Booking | (
                b1.car = b2.car &&
                b1.date.days_passed = b2.date.days_passed &&
                (b1.start_time.minutes_passed >= b2.start_time.minutes_passed &&
b1.end_time.minutes_passed <= b2.end_time.minutes_passed) &&
                (b1.start_time.minutes_passed >= b2.start_time.minutes_passed &&
b1.start_time.minutes_passed <= b2.end_time.minutes_passed) &&
                b1.status = Accepted && b2.status = Accepted && b1.status = Complete &&
b2.status = Complete
        )
}
```

```
fact NoUselessProperties{
	no n: Name |
		(all u: User | n not in u.name)
	no s: Surname |
		(all u: User | s not in u.surname)
	no e: Email |
		(all u: User | e not in u.email)
	no p: Password |
		(all u: User | p not in u.password)
	no t: Time |
		(all o: Occupation | t not in o.starting_time && t not in o.ending_time) and
		(all b: Booking | t not in b.start_time && t not in b.end_time)
	no d: Date |
		(all o: Occupation | d not in o.date) and
		(all b: Booking | d not in b.date)
	no a: Address |
		(all cp: CP | a not in cp.address) and
		(all cpo: CPO | a not in cpo.address)
}

fact NoBookingsAtTheSameTimeForOneCar{
	no disjoint b1, b2: Booking | (
		b1.car = b2.car &&
		b1.date.days_passed = b2.date.days_passed &&
		(b1.start_time.minutes_passed >= b2.start_time.minutes_passed &&
b1.end_time.minutes_passed <= b2.end_time.minutes_passed) &&
		(b1.start_time.minutes_passed >= b2.start_time.minutes_passed &&
b1.start_time.minutes_passed <= b2.end_time.minutes_passed) &&
		b1.status = Accepted && b2.status = Accepted && b1.status = Complete &&
b2.status = Complete
	)
}
//Giving the current time values
```

```
fact ExactValueOfCurrentTime{
        all ct: CurrentTime | (
                (some d: Date | d.days_passed = 3 and ct.date = d) and
                (some t: Time | t.minutes_passed = 3 and ct.time = t)
        )
}

fact ValidStatusesInBooking{
        no b: Booking | (
                all ct: CurrentTime | (
                        (b.date.days_passed < ct.date.days_passed) and
                        (b.status = Accepted)
                ) or
                (
                        (b.date.days_passed > ct.date.days_passed) and
                        (b.status = Complete)
                ) or
                (
                        (b.date.days_passed = ct.date.days_passed) and
                        (b.end_time.minutes_passed <= ct.time.minutes_passed) and
                        (b.status = Accepted)
                ) or
                (
                        (b.date.days_passed = ct.date.days_passed) and
                        (b.end_time.minutes_passed > ct.time.minutes_passed) and
                        (b.status = Complete)
                )
        )
}

fact ValidStatusesInChargingSocket{
        no cs: ChargingSocket | (
                all ct: CurrentTime | (
                        some b: Booking | (
                                (b.status = Accepted) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Free)
                        ) or
                        some b: Booking | (
                                (b.status = Complete) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Free)
                        ) or
                        some b: Booking | (
                                (b.status = Complete) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Occupied)
                        )
                )
        )
}
//predicate
pred show{
        some User &&
        some Booking &&
        some Occupation &&
        some Suggestion &&
        some ChargingSocket &&
        some CP &&
        some CPO &&
        //So that we have only 1 current time value, which has already been defined as 3 days
and 3 mintues passed before
        one CurrentTime
}
run show for 15
```

Result:



52

CPMS:

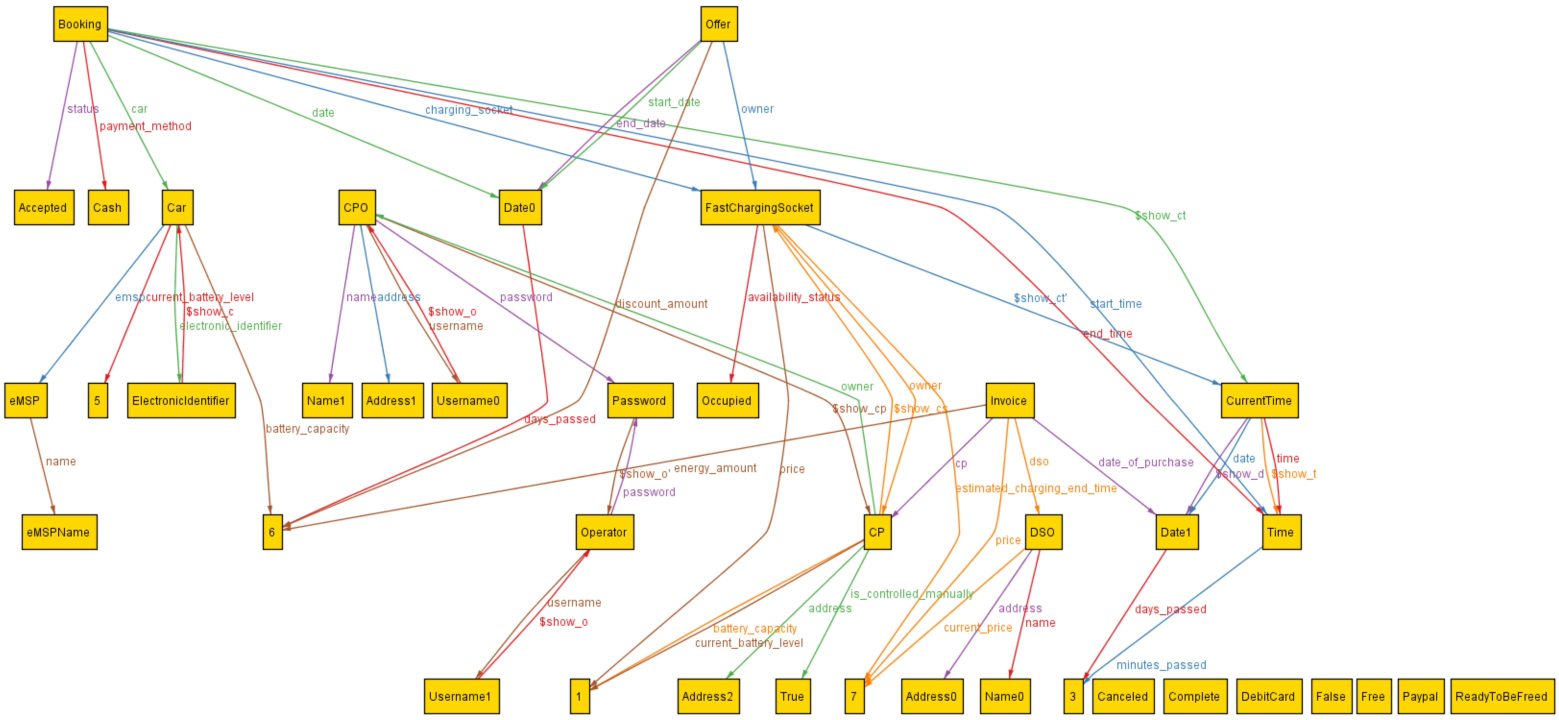
```
//simple signatures
sig Name{}
sig eMSPName{}
sig Username{}
sig Password{}
sig Address{}
sig ElectronicIdentifier{}
sig Condition{}
//Enums
enum PaymentMethod {DebitCard, Paypal, Cash}

enum Boolean {True, False}

enum AvailabilityStatus {Free, Occupied, ReadyToBeFreed}

enum BookingStatus {Canceled, Accepted, Complete, Payed}
//Complex Signatures
sig Date{
        days_passed: Int
}

sig Time{
        minutes_passed: Int
}

sig CurrentTime{
        date: one Date,
        time: one Time
}

sig eMSP{
        name: one eMSPName
}

sig CPO extends Operator{
        name: one Name,
        address: one Address
}

sig Operator{
        username: one Username,
        password: one Password
}

sig DSO{
        name: one Name,
        address: one Address,
        current_price: one Int
}

sig CP{
        owner: one CPO,
        address: one Address,
        battery_capacity: one Int,
        current_battery_level: one Int,
        is_controlled_manually: one Boolean,
        store_energy_in_battery: one Condition,
        use_energy_in_battery: one Condition,
        price_and_offer_strategy: one Condition
}

sig Invoice{
        cp: one CP,
        dso: one DSO,
        energy_amount: one Int,
        date_of_purchase: one Date,
        price: one Int
}

sig Offer{
        owner: one ChargingSocket,
        discount_amount: one Int,
        start_date: one Date,
        end_date: one Date
}

abstract sig ChargingSocket{
        owner: one CP,
        availability_status: one AvailabilityStatus,
        estimated_charging_end_time: one Int,
        price: one Int
}

sig SlowChargingSocket extends ChargingSocket{}

sig FastChargingSocket extends ChargingSocket{}

sig RapidChargingSocket extends ChargingSocket{
        OptimizedModeOn: one Boolean
}

sig Booking{
        car: one Car,
        charging_socket: one ChargingSocket,
        payment_method: one PaymentMethod,
        date: one Date,
        start_time: one Time,
        end_time: one Time,
        status: one BookingStatus
}

sig Car{
        emsp: one eMSP,
        electronic_identifier: one ElectronicIdentifier,
        battery_capacity: one Int,
        current_battery_level: one Int
}

fact ValidValuesForDates{
        all d: Date | d.days_passed > 0
}

fact ValidValuesForTimes{
        all t: Time | t.minutes_passed > 0
}

fact ValidValuesForEstimatedTimeInChargingSockets{
        (all cs: ChargingSocket | cs.estimated_charging_end_time >= 0) and
        (no cs: ChargingSocket | cs.estimated_charging_end_time = 0 and cs.availability_status
= Occupied) and
        (no cs: ChargingSocket | cs.estimated_charging_end_time >0 and
                (cs.availability_status = Free or cs.availability_status = ReadyToBeFreed)
        )
}

fact ValidValuesForPricesInDSOs{
        all dso: DSO | dso.current_price > 0
```

```
}

fact ValidValuesForPricesInInvoices{
        all i: Invoice | i.price > 0
}

fact ValidValuesForPricesInChargingSockets{
        all cs: ChargingSocket | cs.price > 0
}

fact ValidValuesForDiscountAmountsInOffers{
        all o: Offer | o.discount_amount > 0
}

fact ValidValuesForBatteriesInCPs{
        all cp: CP | (
                cp.battery_capacity > 0 &&
                cp.current_battery_level > 0 &&
                cp.current_battery_level <= cp.battery_capacity
        )
}

fact ValidValuesForEnergyAmountInBill{
        all b: Invoice | b.energy_amount > 0
}

fact AllCPOsHaveCP{
        all cpo: CPO | some cp: CP | cpo in cp.owner
}

fact AllCPsHaveChargincSocket{
        all cp: CP | some cs: ChargingSocket | cp in cs.owner
```

```
}

fact AddressesOfCPsAreUnique{
        no disjoint cp1, cp2: CP | cp1.address = cp2.address
}

fact AddressesOfDSOsAreUnique{
        no disjoint dso1, dso2: DSO | dso1.address = dso2.address
}

fact AddressesOfCPOsAreUnique{
        no disjoint cpo1, cpo2: CPO | cpo1.address = cpo2.address
}

fact AddressesOfCPsAndCPOsAreUnique{
        no cp: CP | some cpo: CPO | cp.address = cpo.address
}

fact AddressesOfCPsAndDSOsAreUnique{
        no cp: CP | some dso: DSO | cp.address = dso.address
}

fact AddressesOfDSOsAndCPOsAreUnique{
        no dso: DSO | some cpo: CPO | dso.address = cpo.address
}

fact NamesOfCPOsAreUnique{
        no disjoint cp1, cp2: CPO | cp1.name = cp2.name
}

fact NamesOfDSOsAreUnique{
        no disjoint dso1, dso2: DSO | dso1.name = dso2.name
```

```
}

fact NamesOfCPOsAndDSOsAreUnique{
        no cpo: CPO | some dso: DSO | cpo.name = dso.name
}

fact UniqueElectronicIdentifiers{
        no disjoint c1, c2: Car | c1.electronic_identifier = c2.electronic_identifier
}

fact UniqueUsernames{
        no disjoint o1, o2: Operator | o1.username = o2.username
}

fact UniqueeMSPNames{
        no disjoint e1, e2: eMSP | e1.name = e2.name
}

fact ValidStatusesInBooking{
        no b: Booking | (
                all ct: CurrentTime | (
                        (b.date.days_passed < ct.date.days_passed) and
                        (b.status = Accepted)
                ) or
                (
                        (b.date.days_passed > ct.date.days_passed) and
                        (b.status = Complete)
                ) or
                (
                        (b.date.days_passed = ct.date.days_passed) and
                        (b.end_time.minutes_passed <= ct.time.minutes_passed) and
                        (b.status = Accepted)
                ) or
                (
                        (b.date.days_passed = ct.date.days_passed) and
                        (b.end_time.minutes_passed > ct.time.minutes_passed) and
                        (b.status = Complete)
                )
        )
}

fact ValidStatusesInChargingSocket{
        no cs: ChargingSocket | (
                all ct: CurrentTime | (
                        some b: Booking | (
                                (b.status = Accepted) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Free)
                        ) or
                        some b: Booking | (
                                (b.status = Complete) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Free)
                        ) or
                        some b: Booking | (
                                (b.status = Complete) and
                                (b.date = ct.date) and
                                (b.start_time.minutes_passed <= ct.time.minutes_passed) and
                                (b.end_time.minutes_passed >= ct.time.minutes_passed) and
                                (cs.availability_status = Occupied)
                        )
                    )
                )
}

fact ExactValueOfCurrentTime{
        all ct: CurrentTime | (
                (some d: Date | d.days_passed = 3 and ct.date = d) and
                (some t: Time | t.minutes_passed = 3 and ct.time = t)
        )
}

fact NoUselessProperties{
        no n: Name |
                (all cpo: CPO | n not in cpo.name) and
                (all dso: DSO | n not in dso.name)
        no u: Username |
                (all o: Operator | u not in o.username)
        no p: Password |
                (all o: Operator | p not in o.password)
        no ed: ElectronicIdentifier |
                (all c: Car | ed not in c.electronic_identifier)
        no a: Address |
                (all dso: DSO | a not in dso.address) and
                (all cp: CP | a not in cp.address) and
                (all cpo: CPO | a not in cpo.address)
}

pred show{
        some eMSP and
        some Booking and
        some Operator and
        not one Operator
        some CPO and
        some DSO and
        some Invoice and
        some Offer and
        one CurrentTime
}
run show for 15
```

Result:



5. Effort spent

5.1 Beliakov Maksim

Task	Time spent (h)
Brainstorming and discussion	10
Introduction	4
Overall description	4
Specific requirement	15
Formal analysis	5
Document refinement	1
Total	39

5.2 Aliyev Rustam

Task	Time spent
Brainstorming and discussion	10
Introduction	2
Overall description	4
Specific requirement	8
Formal analysis	14
Document refinement	1
Total	39

5.3 Yizhou Wu

Task	Time spent
Brainstorming and discussion	10
Introduction	2
Overall description	8
Specific requirement	18
Formal analysis	1
Document refinement	2
Total	41

6. References