POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# RASD - Software Engineering 2

## Computer Science and Engineering

Authors:
**Emilio Corigliano (10627041)**
**Federico Mandelli (10611353)**
**Matteo Pignataro (10667498)**

Advisor: Prof. Matteo Camilli
Co-advisors: Prof.ssa Elisabetta Di Nitto, Prof. Matteo Giovanni Rossi
Academic Year: 2022-23

# Abstract

Insert text here

# Contents

# 1. Introduction

## 1.1. Purpose

Due to the recent increase of effort in the battle against climate change, electric vehicles are slowly becoming the new technology for private transport that people use everyday. To sustain this type of strategy, we need to develop a clever and capillary charging system.
eMall is an e-Mobility Service Provider (eMSP) that aims to help the final users dealing with their charging needs by informing the users about the nearby charging stations, their cost and any special offer that they might have. It will allow the users to book/cancel/pay a charge and it will notify the users when the charging process is terminated.
With the integration of the user's calendar, the system will also suggest the best moment in the schedule to charge the vehicle. To have a fully integrated system, all the Charging Point Operators (CPOs) will have a technological support called CPMS to interface the service with the physical charging stations and to manage all the energy sources like batteries and Distribution System Operators (DSOs). CPMSs will be in charge of deciding the energy source and, in case of batteries in a charging station, they will also manage their charging.

### 1.1.1. Goals

**G1** The eMSP shall help the user to select the station;

**G2** The eMSP shall allow the user to book/pay/cancel a charge;

**G3** The eMSP shall allow the user to perform a charge;

**G4** CPMSs shall handle the vehicle charging cycles;

CPMS o charging stations

**G5** CPMSs shall manage the charging stations;

## 1.2. Scope

**W1** People charge electric vehicles in different modes (NORMAL, FAST, SUPER-FAST); [**G1**, **G2**, **G3**, **G4**]

**W2** People use web calendar; [**G1**, **G2**, **G3**]

**W3** People pay for the charging service; [**G3**]

Anche G3?

**W4** DSOs supply energy to CPOs; [**G5**]

**W5** Some CPOs own batteries; [**G4**, **G5**]

**W6** CPOs decide whether to use batteries or DSO supplied energy; [**G4**, **G5**]

**S1** The eMSP suggests the user to charge the vehicle; [**G1**]

**S2** The eMSP notifies the user when the charging process is finished; [**G3**]

**S3** CPMSs acquire information about energy prizes from DSOs; [**G5**]

**S4** The user books a charge using the eMSP; [**G2**]

**S5** The user asks the eMSP for suggestions about charging station; [**G1**]

**S6** The user pays for the service using the eMSP; [**G3**]

**S7** CPOs gather the energy source through the CPMS; [**G4**, **G5**]

## 1.3. Definitions, Acronyms, Abbreviations

### 1.3.1. Definitions

## Glossary

**Partita IVA** Outside the Italian territory it corresponds to the VAT number. It is a unique identifier for the operators that want to perform an economical activity in the national territory. 8, 9, 11, 14, 39

### 1.3.2. Acronyms

| | |
|---|---|
| **eMSP** | e-Mobility Service Provider |
| **CPO** | Charging Point Operator |
| **CPMS** | Charge Point Management System |
| **DSO** | Distribution System Operator |
| **eMall** | e-Mobility for All |
| **API** | Application Programming Interface |
| **RACS** | Reliable Array of Cloned Services |
| **RAPS** | Reliable Array of Partitioned Services |
| **GDPR** | General Data Protection Regulation |
| **SoC** | State of Charge |
| **GPS** | Global Positioning System |

## 1.4. Revision history

## 1.5. Reference Documents

## 1.6. Document Structure

The document is divided in six main sections:

- **Introduction**: The introduction illustrates the problem to the reader and enumerates all the goals that the system needs to achieve. There are also more formal descriptions about the world (world phenomena) and the interactions between the system and the world (shared phenomena). At the end of the introduction there is a reference subsection for definitions, revision history and reference documents;

- **Overall Description**: It is an high level description of the dynamic interaction between stakeholders and the system. For this reason in this section there are the main scenarios descriptions and a UML diagram which specifies all the relations from an upper model perspective. There is a subsection that illustrates the fundamental requirements of the system and another which specifies the type and description of any user. At the end of this section there is a collection of assumptions that are made over the complete project;

- **Specific Requirements**: This section focuses on all the details introduced in the **Overall Description**, it formalizes all the requirements about the system and all the scenarios. For this reason, use case and sequence diagrams are illustrated. More constraints on the performance, design aspects and attributes of the software are shown;

- **Formal Analysis with Alloy**: It represents a formal description in Alloy language of the problem, with some formal constraints that need to be satisfied (asserts). This formalization is useful to validate the model itself and to verify that all the assertions are granted.

- **Effort Spent**: Summarizes the total hours spent on the document formalization;

- **References**: Summarizes all the references documents that we used during the description.

# 2. Overall Description

## 2.1. Product perspective

### 2.1.1. Scenarios

It is assumed that in **S3**,**S4**,**S5**,**S6**,**S8**, the user is already logged in the system (**S2**). In **S11** and **S13** we assume that the CPOmaintainer is already logged in the CPMS.

**S1** User Signs up:
Lucy, wanting to use the system, opens the app, she is prompted to login or register, she chooses to register herself and inserts her personal info (email, password, birthday, payment information, vehicle info); an email is sent with a link to confirm the activation of the account, if the link is clicked within the first 15 minutes the account is activated and the sign up is successful, otherwise it is considered failed and the process must be repeated;

**S2** User Logs in:
Jay, after signing up, opens the app and he is prompted to insert his email and password, if the given information are correct the login is successful and he obtains access to his account and the services of the app, otherwise the login is unsuccessful and it must be repeated;

**S3** User searches for stations:
Robert, opens the app, inserts location and time frame to search for charging stations. Once submitted, a list of available charging stations is displayed, the list is ordered by the distance of the station from the desired location. Via a menu, Robert can choose to order the stations either via distance, price or charging type (super-fast, fast, normal); he can also set to display unavailable stations and set the maximum distance from the chosen location;

**S4** User books a charge:
Jessica, after choosing a station, decides to book a charge selecting the desired time slot. Station location and booked time frame are displayed and she is asked to confirm the booking via a popup. She receives a confirmation email with the details of the charge (Location, time frame, socket id) and a confirmation pin to insert at the station;

**S5** User pays a charge:
John, after booking a charge, has to pay it before actually performing it. He selects the wanted charge and proceeds with the payment. He than receives an email that summarizes the payment details;

**S6** User cancels a charge:
Luke, after booking a charge, wants to cancel it. He opens the app, selects the booking he wants to cancel and confirms the action. A popup appears asking confirmation: if it is pressed the booking is removed, the station returns available, a refound is issued and a confirmation email is sent to the user; otherwise the booking is still valid;

**S7** User charges the vehicle:
Mary, after booking a charge, arrives at the station, she parks her vehicle at the de-

signed socket and plugs her vehicle in. Mary then inserts the confirmation pin in the socket to start the charge. The socket displays on a monitor the status and the finishing time of the charge. Once the charge is finished Mary receives a notification, she gets her vehicle and completes the charge;

**S8** User gets charging suggestion based on his calendar:
Josh is a very busy man and also an avid web calendar user, setting up every event with correct time and location. The service accessing his calendar finds the closest available charging station for each vehicle movement, it connects to the vehicle while driving and stores the last charge level. Once the battery is below fifty percent Josh gets notified about the possibility to charge his vehicle in an available time slot near his location. Josh liking the idea opens the app and he confirms the booking;

**S9** CPO subscribes to the system:
Judy, the CEO of a famous CPO, wants to subscribe it to eMall to improve sales. She opens the eMall website and selects to sign up as CPO, she inserts the name, email, Partita IVA, a master password and connects the CPMSs to the site via an Application Programming Interface (API) reference;

**S10** CPOmaintainer logs into his assigned CPMS:
Brett a CPO employee wants to access the service, he connects to the CPMS and inserts the ID and password, if correct he logs in; otherwise the procedure fails and must be repeated;

**S11** CPOmaintainer adds stations to the CPMS:
Frank, the responsible for a CPMS, wants to add stations to the CPMS in preparation of subscribing to eMall. For each station he has to insert the API reference, wether to use the CPMS automatic source selector or to choose the preferred energy source;

**S12** CPOmaintainer updates settings and strategy about his CPMS:
Andy, after logging in has access to his CPMS. Here he can change the energy source and create maintainer accounts inserting the ID and password;

**S13** CPOmaintainer manages his assigned CPMS
Lisa, a maintainer at a cpo logs in the service, here she can see the info of each station of the CPMS assigned to her. For each station she can: check the status(functioning or not), choose the energy source, monitor the consumes, profitability and the usage of a specified station.

### 2.1.2. UML diagram

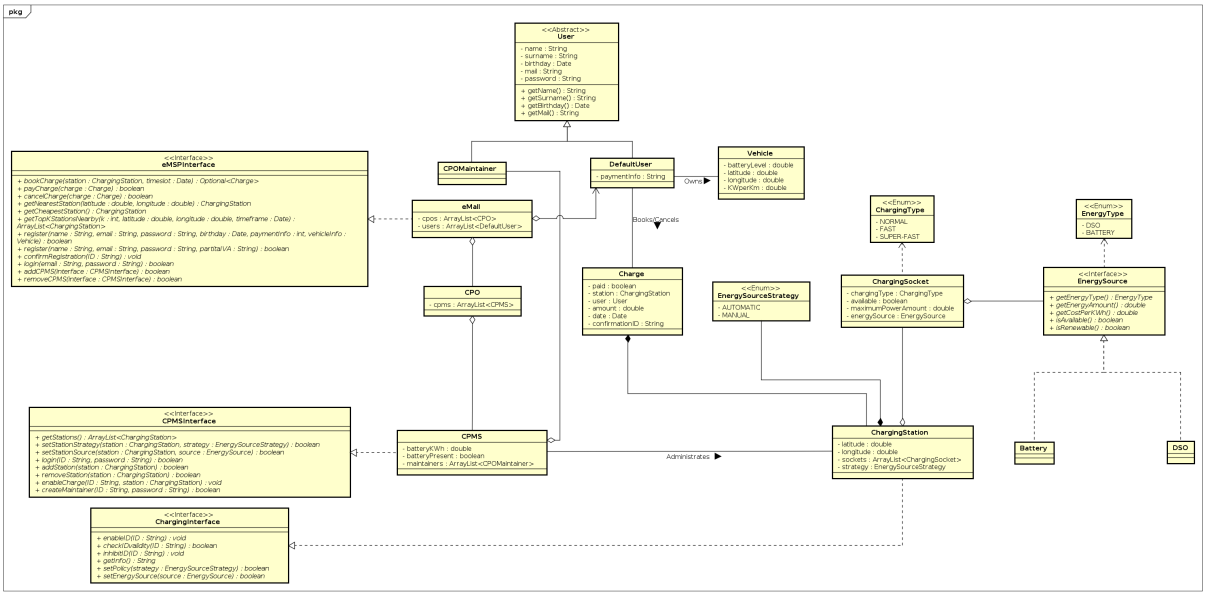

pkg
<<Abstract>>
User
- name : String
- surname : String
- birthday : Date
- mail : String
- password : String
+ getName() : String
+ getSurname() : String
+ getBirthday() : Date
+ getMail() : String
<<Interface>>
eMSPInterface
+ bookCharge(station : ChargingStation, timeslot : Date) : Optional<Charge>
+ payCharge(charge : Charge) : boolean
+ cancelCharge(charge : Charge) : boolean
+ getNearestStation(latitude : double, longitude : double) : ChargingStation
+ getCheapestStation() : ChargingStation
+ getTopKStationsNearby(k : int, latitude : double, longitude : double, timeframe : Date) : ArrayList<ChargingStation>
+ register(name : String, email : String, password : String, birthday : Date, paymentInfo : int, vehicleInfo : Vehicle) : boolean
+ register(name : String, email : String, password : String, partitalVA : String) : boolean
+ confirmRegistration(ID : String) : void
+ login(email : String, password : String) : boolean
+ addCPMS(interface : CPMSInterface) : boolean
+ removeCPMS(interface : CPMSInterface) : boolean
CPOMaintainer
DefaultUser
- paymentInfo : String
Owns
Vehicle
- batteryLevel : double
- latitude : double
- longitude : double
- KWperKm : double
eMall
- cpos : ArrayList<CPO>
- users : ArrayList<DefaultUser>
Books/Cancels
<<Enum>>
ChargingType
- NORMAL
- FAST
- SUPER-FAST
<<Enum>>
EnergyType
- DSO
- BATTERY
CPO
- cpms : ArrayList<CPMS>
Charge
- paid : boolean
- station : ChargingStation
- user : User
- amount : double
- date : Date
- confirmationID : String
<<Enum>>
EnergySourceStrategy
- AUTOMATIC
- MANUAL
ChargingSocket
- chargingType : ChargingType
- available : boolean
- maximumPowerAmount : double
- energySource : EnergySource
<<Interface>>
EnergySource
+ getEnergyType() : EnergyType
+ getEnergyAmount() : double
+ getCostPerKWh() : double
+ isAvailable() : boolean
+ isRenewable() : boolean
<<Interface>>
CPMSInterface
+ getStations() : ArrayList<ChargingStation>
+ setStationStrategy(station : ChargingStation, strategy : EnergySourceStrategy) : boolean
+ setStationSource(station : ChargingStation, source : EnergySource) : boolean
+ login(ID : String, password : String) : boolean
+ addStation(station : ChargingStation) : boolean
+ removeStation(station : ChargingStation) : boolean
+ enableCharge(ID : String, station : ChargingStation) : void
+ createMaintainer(ID : String, password : String) : boolean
CPMS
- batteryKWh : double
- batteryPresent : boolean
- maintainers : ArrayList<CPOMaintainer>
Administrates
ChargingStation
- latitude : double
- longitude : double
- sockets : ArrayList<ChargingSocket>
- strategy : EnergySourceStrategy
Battery
DSO
<<Interface>>
ChargingInterface
+ enableID(ID : String) : void
+ checkIDvalidity(ID : String) : boolean
+ inhibitID(ID : String) : void
+ getInfo() : String
+ setPolicy(strategy : EnergySourceStrategy) : boolean
+ setEnergySource(source : EnergySource) : boolean

**Figure 1:** UML

## 2.2. Product functions

In the following subsections the functions of each subsystem are described.

### 2.2.1. eMall

**Accessing the eMall** In order to access the system features an authentication is required. When registering it's mandatory for users to insert Name, Surname, birthday, e-Mail, Password and a Payment Method whereas for CPOs the required information are Name, e-Mail, Password and Partita IVA. For the login, an authentication with e-Mail and password is required.

**Performing a charge** The main feature of the system is to help people booking a charge efficiently. The system allows people to see the availability of charging stations and choose where and when to charge the vehicle. If a user changes his mind, he can delete a previously booked charge. When the user arrives to the booked socket, he has to insert the confirmation pin in order to start the charging process. The system notifies the user when the charging process is completed.

**Showing information about charging stations** Whenever a user selects a charging station, location, price, a parameter on how green the energy provided is, special offers and availability of sockets in the station are shown.

**Suggesting recharge of the vehicle** The system offers proactive suggestions about the vehicle recharge via the connection between the vehicle, the electronic calendar and eMall.

It is able to suggest the user when and where to charge the vehicle to minimize the cost, the environment impact, and the distance from the scheduled appointments.

### 2.2.2. CPMS

**Accessing the CPMS** In order to manage the CPMS an authentication with proper authorization is required. So a CPOsmaintainer can login with ID and password.

**Managing the energy source for a charging station** An authorized CPOmaintainer can choose manually the energy source(battery or DSO) for each station. Thus a CPOsmaintainer can decide the cost of a charge and create special offers to increase visibility of the station or to promote greener solutions. If the CPOmaintainer wishes, the CPMS can also work in automatic mode, so the system is able to make all the decisions autonomously.

## 2.3. User characteristics

We consider the following actors in the eMall system:

**A1** **Unregistered user:** A user that needs to register before accessing all the eMall or CPMS services;

**A2** **Registered default user:** A user that has access to all the eMall features. This actor is associated to an electric vehicle and can visualize the nearest stations, book/cancel/-pay a charge, visualize the status of a charge and activate the automatic suggestion service based on the agenda;

**A3** **Registered CPO maintainer user:** A user that has access to all the CPMS features. These features allow the maintainer to configure the CPMS depending on the energy source strategy and visualize all charging stations statuses;

## 2.4. Assumptions dependencies and constraints

### 2.4.1. Assumptions

**DA1** Users insert genuine data in the forms;

**DA2** Users(Including CPOs) do not use the system with malicious intent;

**DA3** All the electric vehicles can be charged by all the stations (no incompatibility);

**DA4** All the user have an active internet and GPS connection always available while using the service;

**DA5** Once installed a CPMS the first login is done using default user and password and a a first CPOmaintainer is configured.

### 2.4.2. Constraint

**C1** If a User wants to change the time slot of a charge he is required to cancel and re-book the charge;

# 3. Specific Requirements

## 3.1. External interfaces requirements

### 3.1.1. User interfaces

**eMSP** The eMSP should be accessible to the user through an application installed on the mobile device. The first interface shown to the user, if not already done, is the *login* page where the user have to input the username (or email) and password in order to authenticate. From the *login* page there is also the the possibility to go to the *register* page where the fields for inserting the necessary information are present. After logging in, there are multiple views available to the user, corresponding of multiple tabs in the app, which are:

- A satellite map of the charging stations near the position of the user;
- A ranked list personalizable based on parameters chosen by the user (distance, price, environmental friendliness. . . ).
- A screen for enabling/disabling suggestions from the system, setting up the connection to the user online calendar.

Selecting a station (from the ranked list or the satellite map) the specific information about the station are shown. The user can select the date and the time-slot from the available ones.

A CPO can be registered on the system with a special registration form providing Company name, password and Partita IVA. Once the CPO is registered and logged in, he can insert the reference to the interface of a CPMS.

**CPMS** The CPMS works as a web application; A CPOmaintainer accesses directly to the CPMS and has the possibility to handle all the stations of the CPO.

Once accessed to the system, the CPOmaintainer has the possibility to view the system status, seeing the list of charging stations with their policy, the available sockets and the State of Charge (SoC) of batteries. Selecting a socket the schedule of the booked charges can be viewed. The CPOmaintainers can change the policy of each charging station; in particular they can:

- Choose a particular energy provider from a list of DSOs with the optional opportunity to recharge charging station batteries with this DSO;
- Choose to use only the energy stored in the batteries and, when this ends, go into automatic mode;
- Choose the automatic mode, which will choose autonomously which DSO to use, when to use batteries and when to recharge them;

The CPO can also view the price of his service and set the revenue percentage for a single charge.

Check with assignment document for all the functions

### 3.1.2. Hardware interfaces

Aggiorna Requirements con spiegazione di EnergySourceStrategy

**eMSP** The user, in order to interact with the eMSP, must have a device that is provided with a Global Positioning System (GPS) and internet connection. Thanks to this, the user can search for close charging stations, see if those are available and can book or cancel a charge. A Bluetooth peripheral should also be available to the user when he is in the vehicle, in order to make a connection with it. Thanks to this the device can query the vehicle infos (such as average battery consumption per kilometer, estimated autonomy and SoC) so that the system can suggest to the user when and where to charge the vehicle.

**CPMS** In order to use the CPMS, the CPOmaintainer (the only type of user of this system) should have a personal computer with internet connection available so that it's possible to see the system info and communicate changes to the system (i.e. change the energy source of a charging station or setting the new revenue for a charge).

Check that we will always set the revenue instead of the final price in order to be consistent

**Charging socket** The Charging sockets should have a pad for inserting the pin that the user have to validate the charge.

Add "We assume that Charging sockets have internet connection and an appropriate interface" to the assumptions

### 3.1.3. Software interfaces

**eMSP** The eMSP does not provide any software interface because no external software should query this system.

Add "The software utilizes payment APIs" to the assumptions

**CPMS** The CPMS should provide to the external world interfaces for:

- Book the charges in a particular time-slot (accepting also a *chargeID*, a PIN in order to authorize the charge once the user gets in the station);
- Get information of a particular charging station (location, price of the charge, parameter of environmental friendliness, type of charges available);
- Get the availability state of a particular socket;
- Get the future availability of the sockets managed by the system;

Compara software interfaces del CPMS con sequence per scoprire tutte le interfacce

### 3.1.4. Communication interfaces

**eMSP** The eMSP should use internet connection in order to interact with the back-end of the system, query the different CPMSs and be connected to the electronic calendar. In order to communicate with the vehicle the user device should also be provided with bluetooth so that can retrieve data from the vehicle and use that for suggesting when and where to charge the vehicle.

**CPMS** The eMSP should be provided with a local connection in order to link all the infrastructure and make it managable by a user in the local connection. An internet connection should also be present in order to make the system reachable by the external world; in particular it is needed for queries and exteral functions made by users (like booking a charge, canceling a charge, seeing what timeslots are available) and in order to manage remotely the system from the CPOmaintainers.

## 3.2. Functional requirements

**R1** The eMSP shall allow the users to register, providing name, surname, birthday, email, password, payment method;

**R2** The eMSP shall allow the user to login with email and password;

**R3** The eMSP shall provide informations about a selected station like types of sockets available, price for the charge, location, available timeslots;

**R4** The eMSP shall reserve a socket in the right charging station with the wanted type of charge for a user who registered for a charge through the application;

**R5** The eMSP shall allow only one user to book a socket in a particular time slot, so no booking collisions are allowed;

**R6** The eMSP shall take the service money from the user when the time slot is booked;

**R7** The eMSP shall refund the user when a charge is canceled;

**R8** The eMSP shall allow the user to see nearby[1] charging stations ordered by distance, price or environmental friendliness;

**R9** The eMSP shall be able to connect to a calendar, retrieve informations about the appointments and parse them;

**R10** The eMSP shall be able to use the informations about the appointments, the charging stations and the vehicle in order to proactively suggest to the user when and where to charge the vehicle;

**R11** The eMSP shall notify the user when the charging process is finished via a notification;

[1] This parameter may be setted by the user

**R12** The eMSP shall be able to communicate with different CPOs;

**R13** The eMSP shall allow a CPO to register, providing name, email, password, Partita IVA;

**R14** the eMSP shall allow to add to an already registered CPO a CPMS, providing connection to the CPO, number of charging sockets, type of charges;

**R15** The eMSP shall verify the correctness of the identification data for the CPOs;

**R16** The CPMS shall be reachable by eMSPs in order to perform/cancel a booking or query the system for retrieving informations;

**R17** The CPMS shall allow the CPO to modify the informations about their systems, such as adding/removing charging stations, adding/removing charging sockets, modifying the type of charge the sockets have, modify the availability/quantity of batteries, adding/removing possible DSOs.

**R18** The CPMS shall allow the CPOmaintainer to access to the system;

**R19** The CPMS shall allow the CPOmaintainer to set the revenue wanted;

**R20** The CPMS shall allow the CPOmaintainer to set special offers;

**R21** The CPMS shall allow the CPOmaintainer to choose the charging mode for a particular charging station (automatic, specific DSO, cheapest DSO, most environmental friendly DSO, use of batteries);

**R22** The CPMS shall allow the CPOmaintainer to choose in manual mode whether to charge the batteries;

What if a user finishes prematurely the charge? will he be refunded by the time of charge left?

### 3.2.1. Use case diagrams

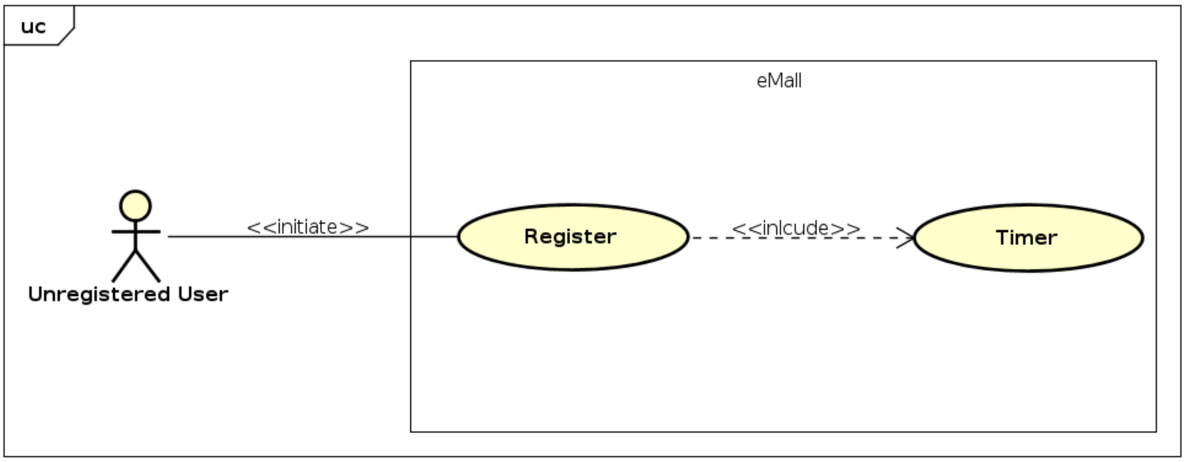


**Figure 2:** Unregistered user use case

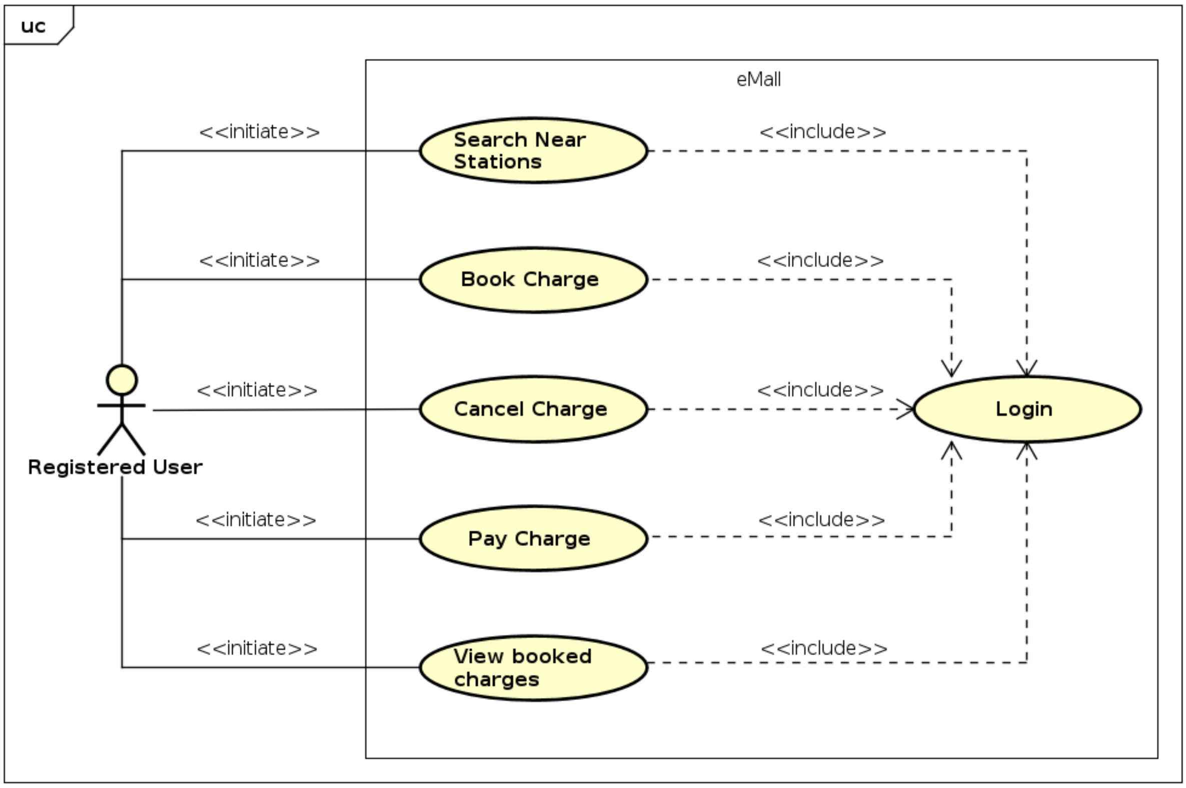


**Figure 3:** Registered user use case

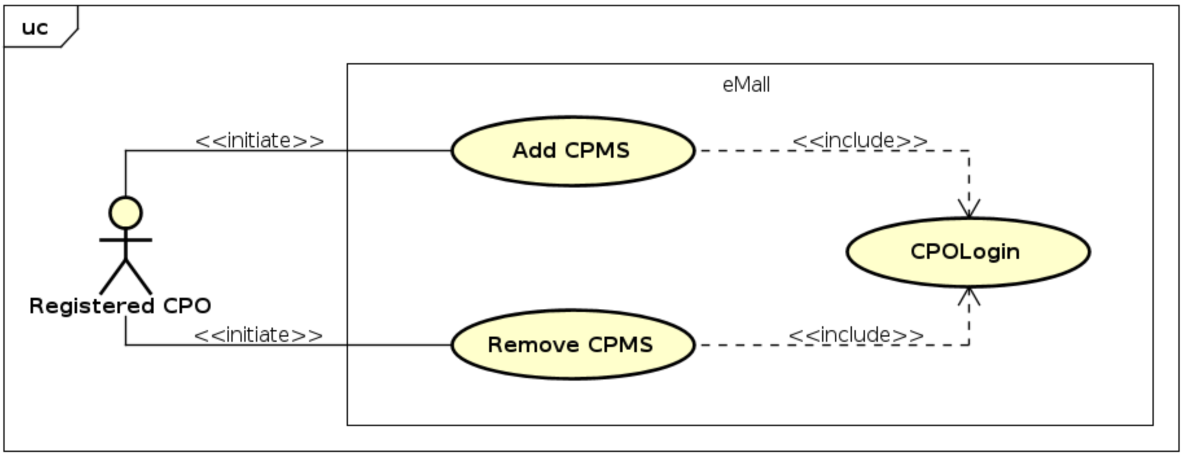


**Figure 4:** Registered CPO

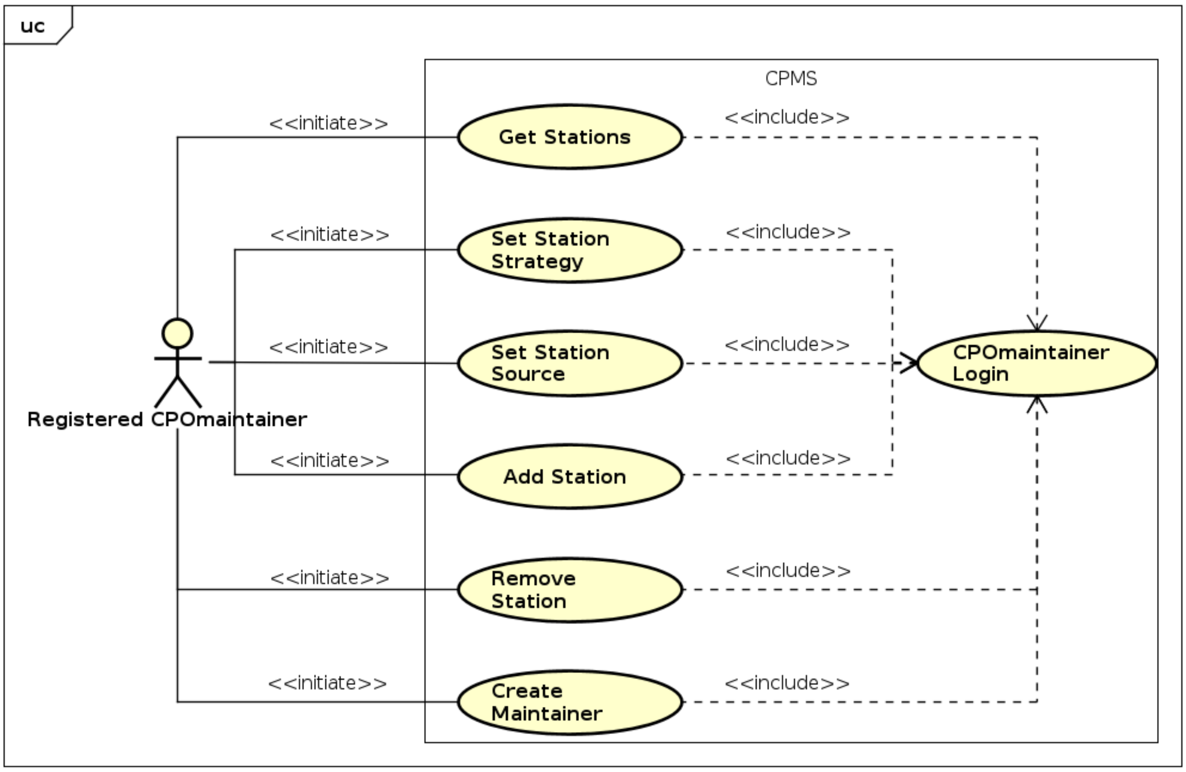


**Figure 5:** Registered CPOmaintainer

### 3.2.2. Sequence diagrams

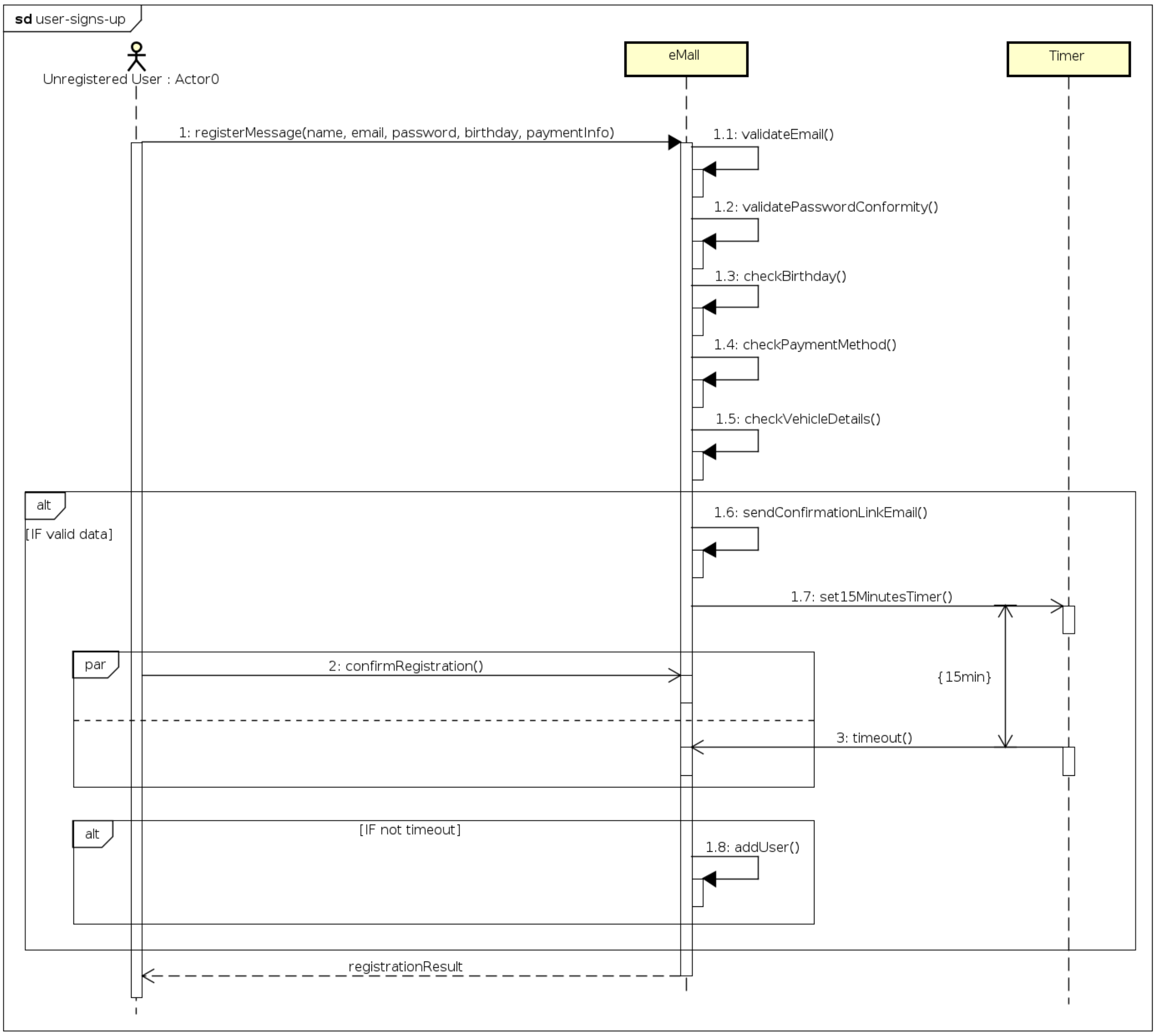

sd user-signs-up
Unregistered User : Actor0
eMall
Timer
1: registerMessage(name, email, password, birthday, paymentInfo)
1.1: validateEmail()
1.2: validatePasswordConformity()
1.3: checkBirthday()
1.4: checkPaymentMethod()
1.5: checkVehicleDetails()
alt
[IF valid data]
1.6: sendConfirmationLinkEmail()
1.7: set15MinutesTimer()
par
2: confirmRegistration()
{15min}
3: timeout()
alt
[IF not timeout]
1.8: addUser()
registrationResult

**Figure 6:** Registration into eMall sequence

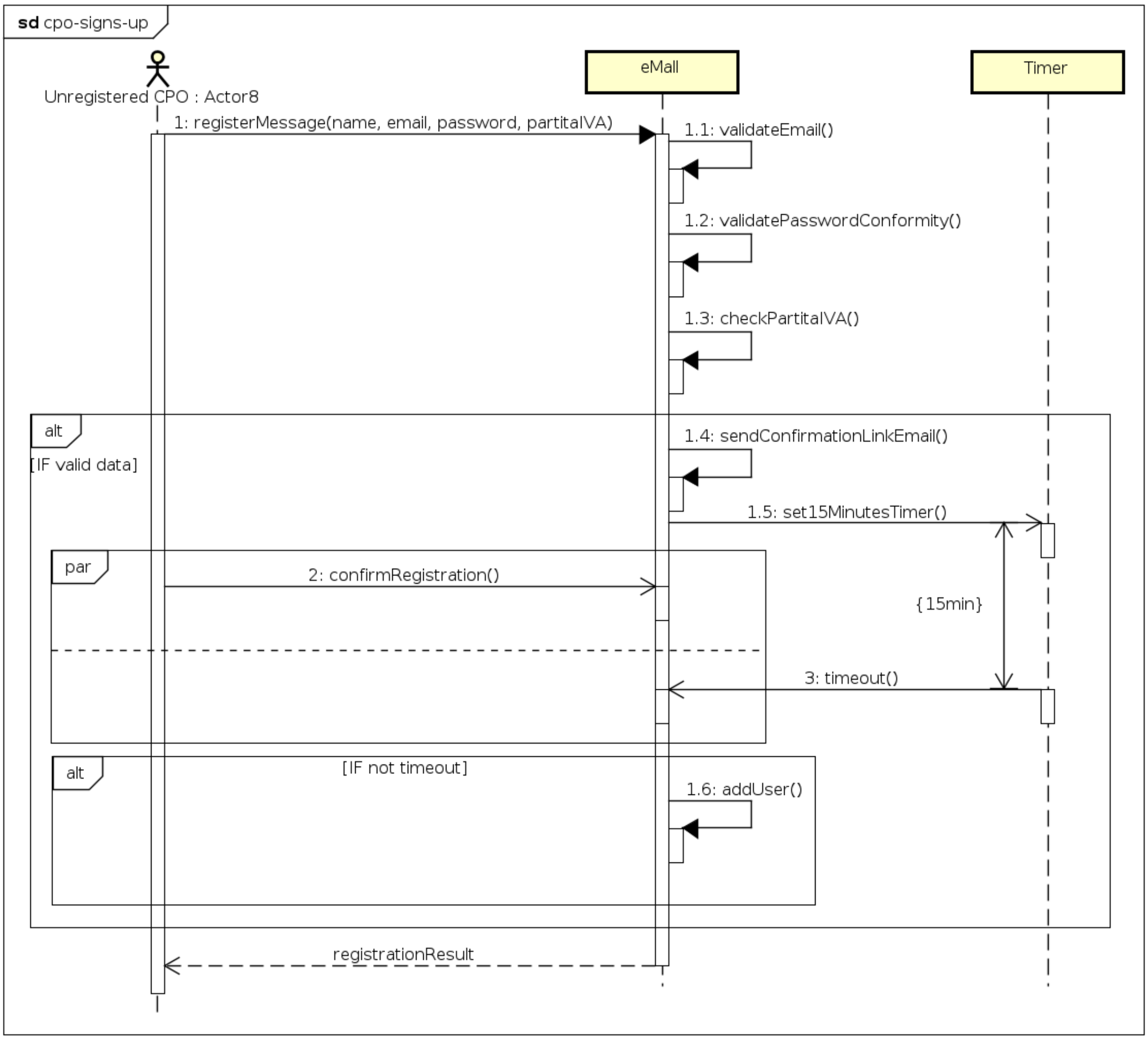


**Figure 7:** Registration of CPO into eMall sequence

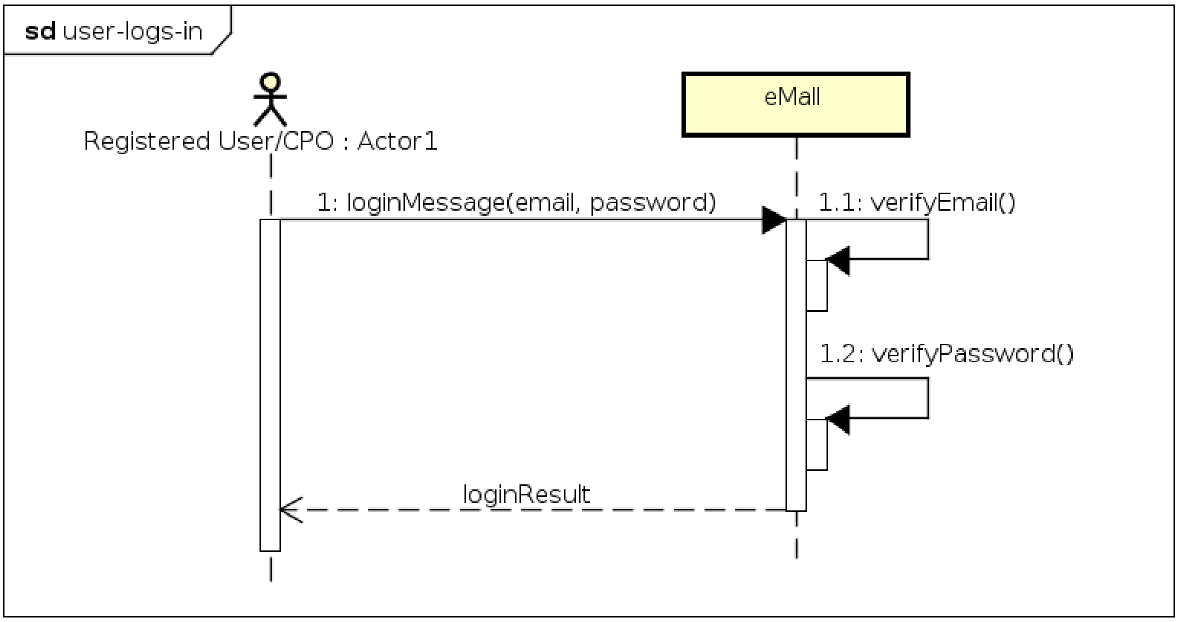


**Figure 8:** Login into eMall sequence

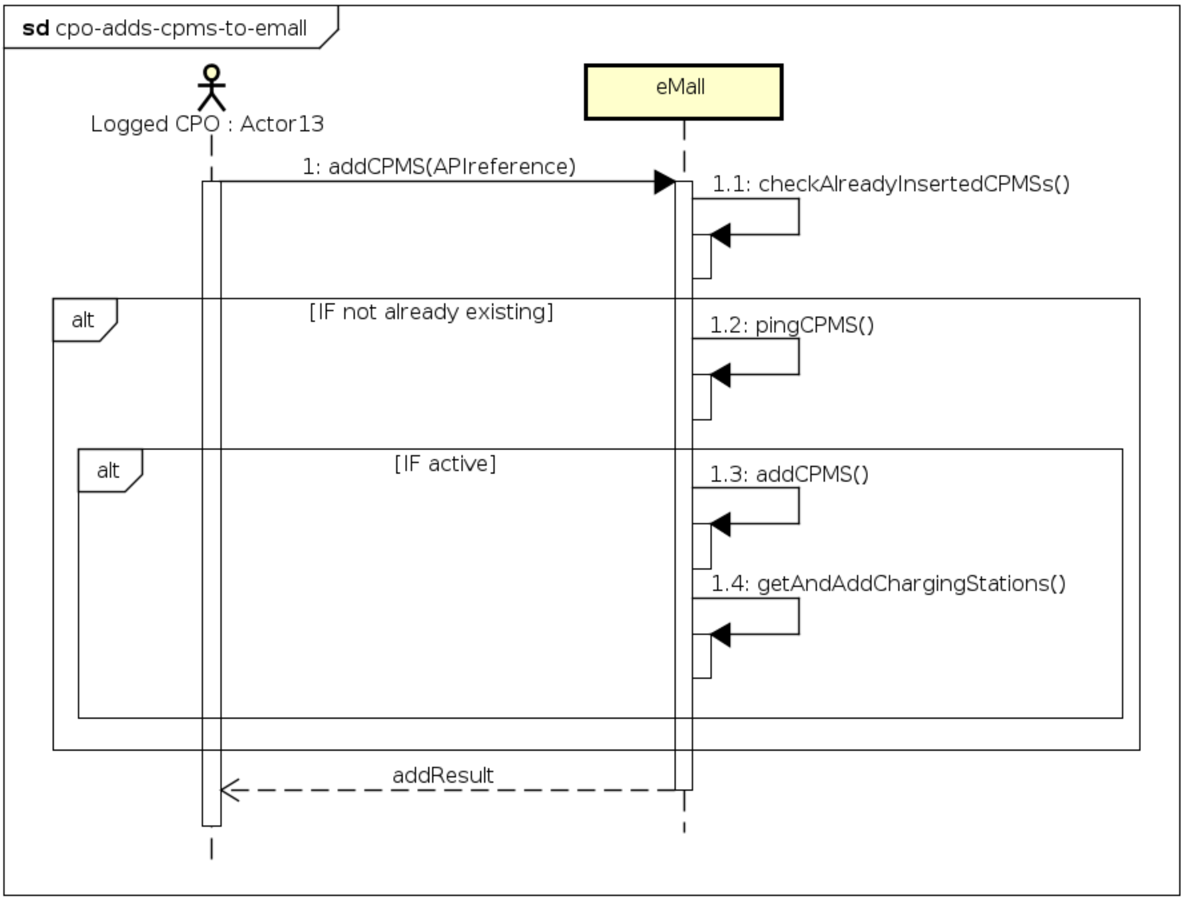

sd cpo-adds-cpms-to-emall
Logged CPO : Actor13
eMall
1: addCPMS(APIreference)
1.1: checkAlreadyInsertedCPMSs()
alt
[IF not already existing]
1.2: pingCPMS()
alt
[IF active]
1.3: addCPMS()
1.4: getAndAddChargingStations()
addResult

**Figure 9:** CPO adds a CPMS into eMall

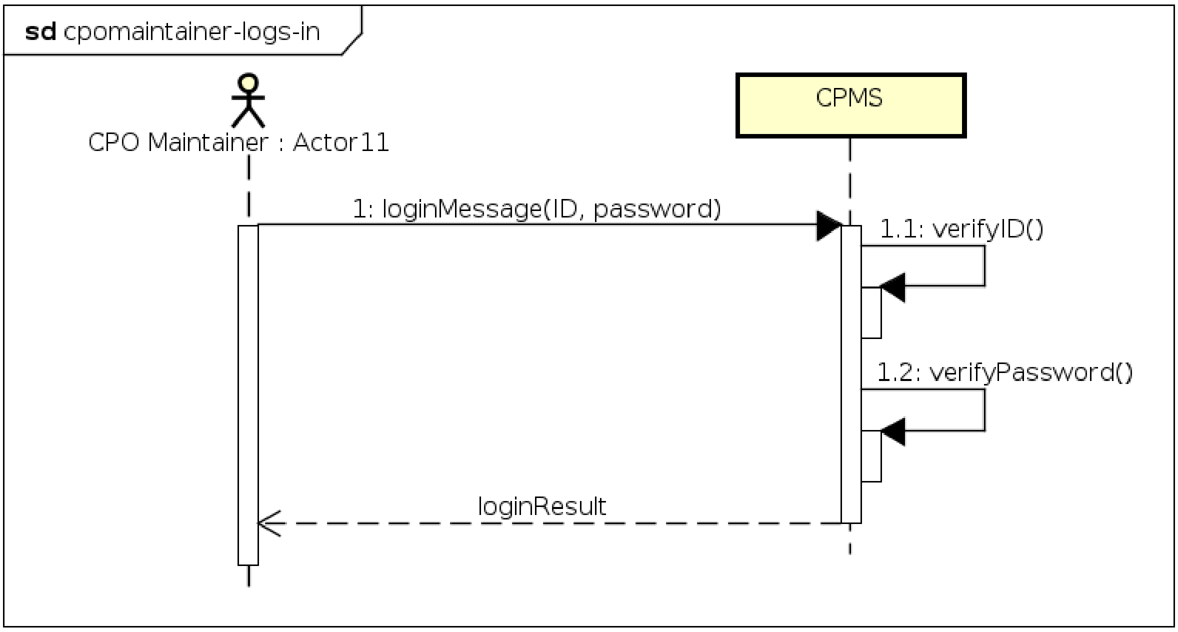


**Figure 10:** CPO maintainer logs into CPMS

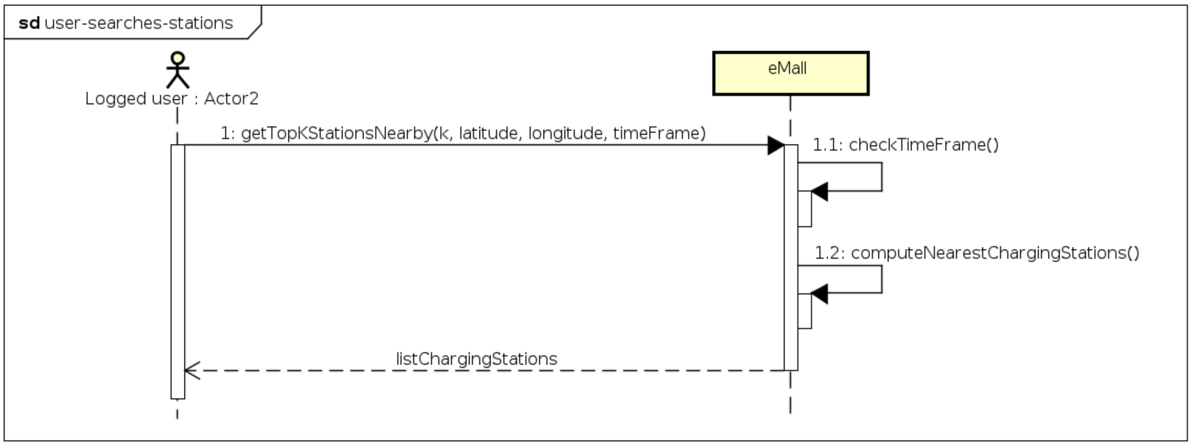

sd user-searches-stations
Logged user : Actor2
eMall
1: getTopKStationsNearby(k, latitude, longitude, timeFrame)
1.1: checkTimeFrame()
1.2: computeNearestChargingStations()
listChargingStations

**Figure 11:** Get the nearby charging stations

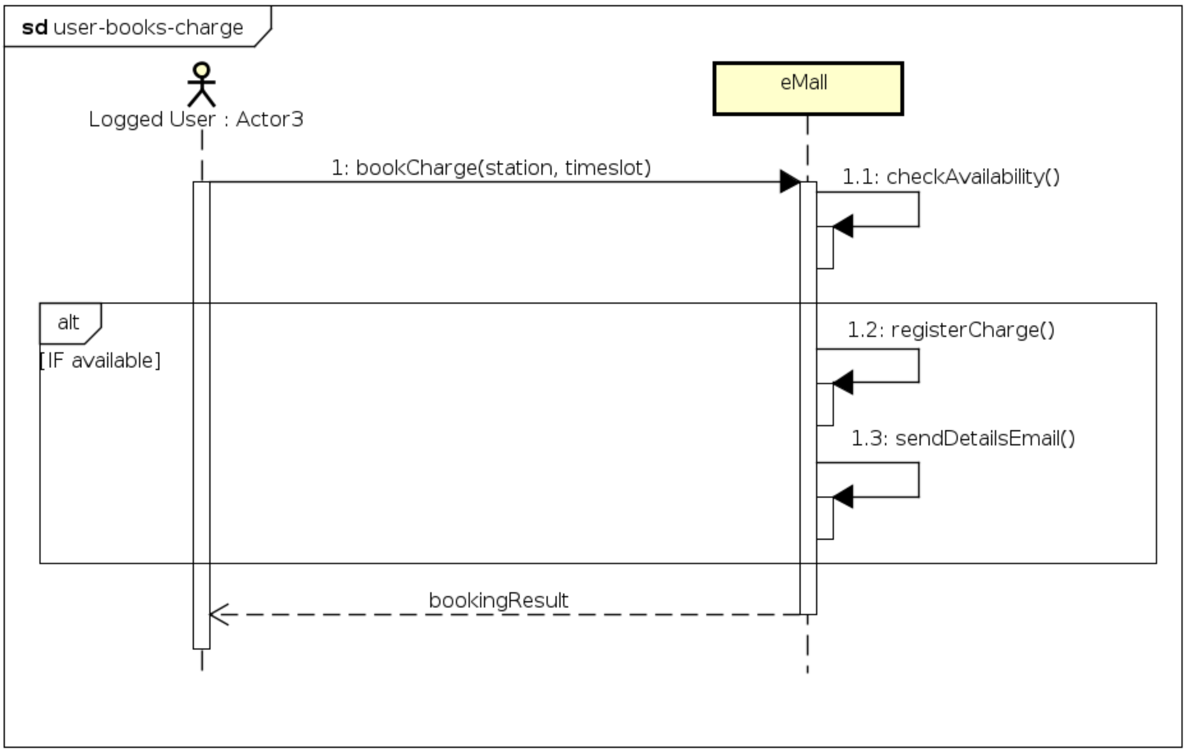


**Figure 12:** Book a charge sequence

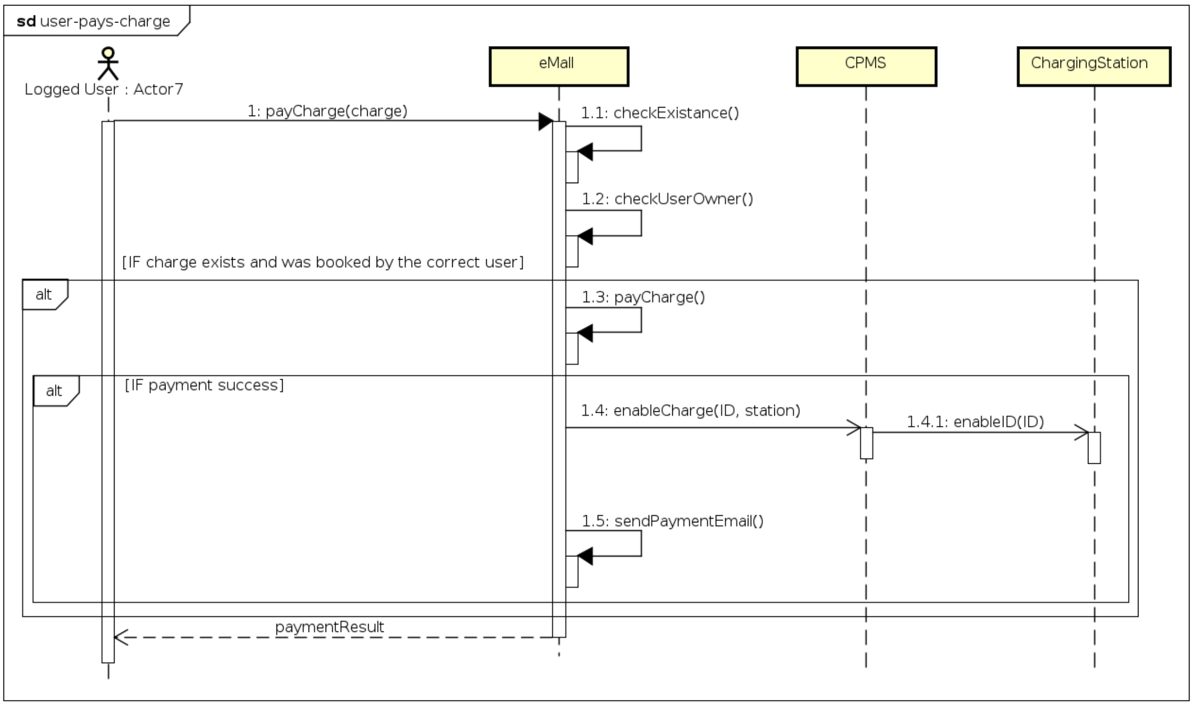


**Figure 13:** Pay a charge sequence

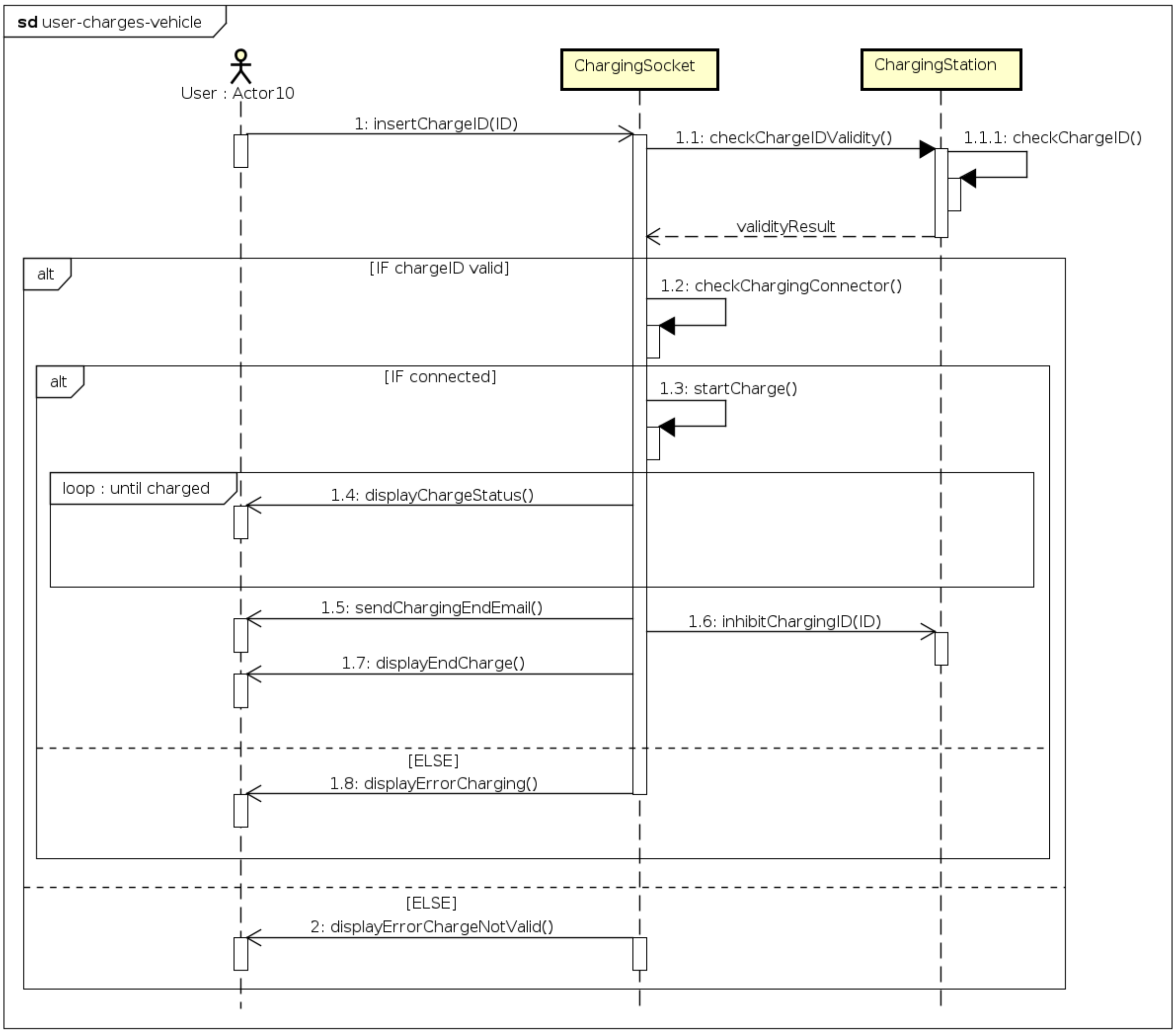


**Figure 14:** Perform a charge sequence

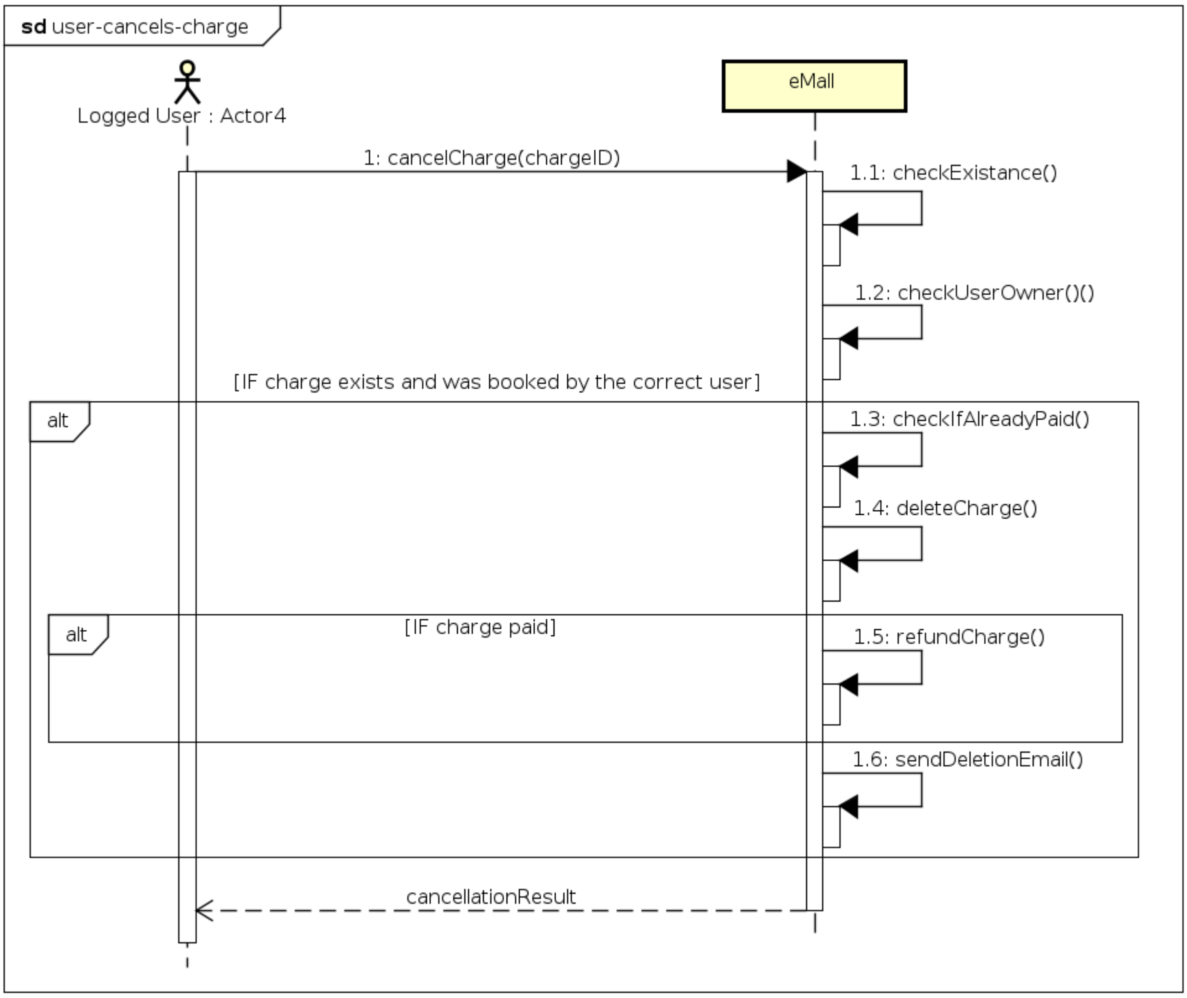


**Figure 15:** Cancel a charge sequence

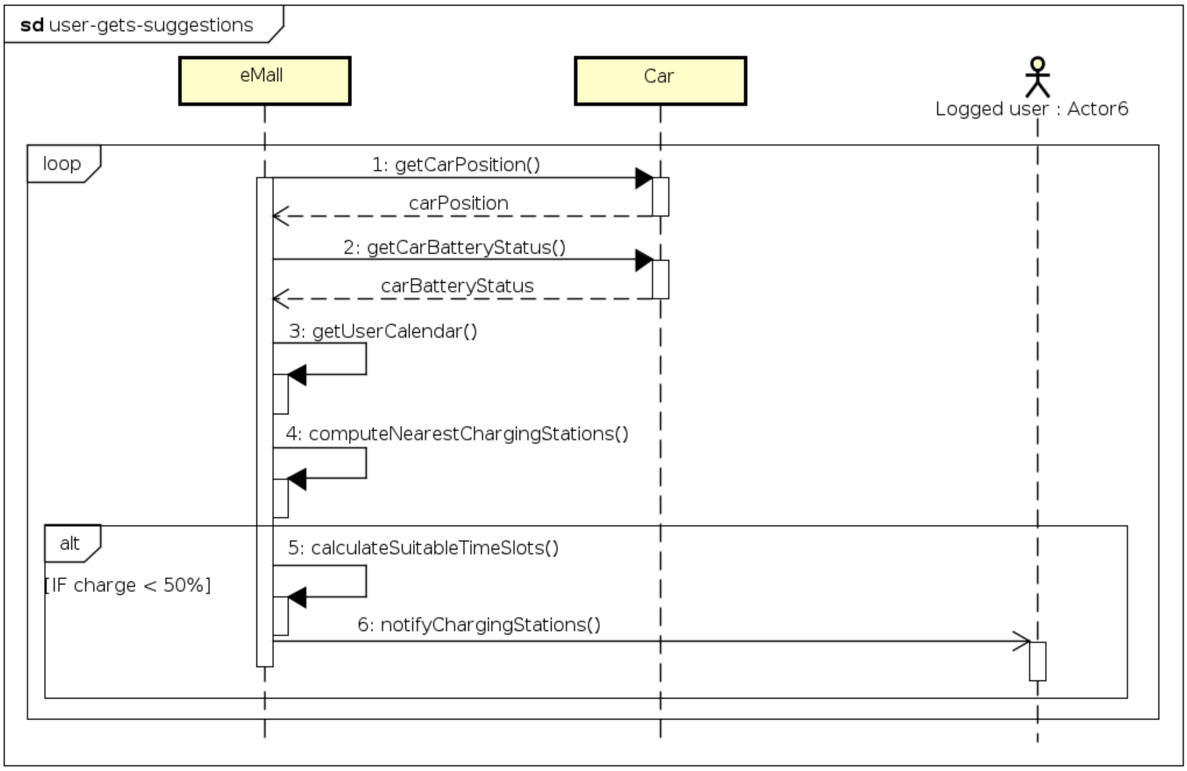


**Figure 16:** Charging suggestions via calendar sequence

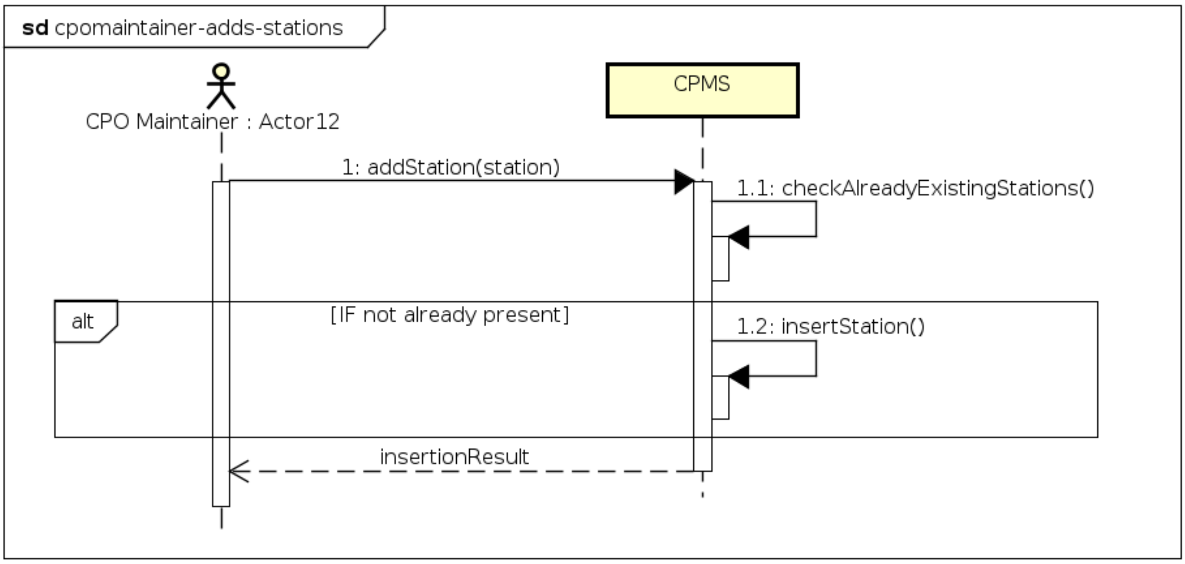


**Figure 17:** CPO maintainer adds stations to CPMS

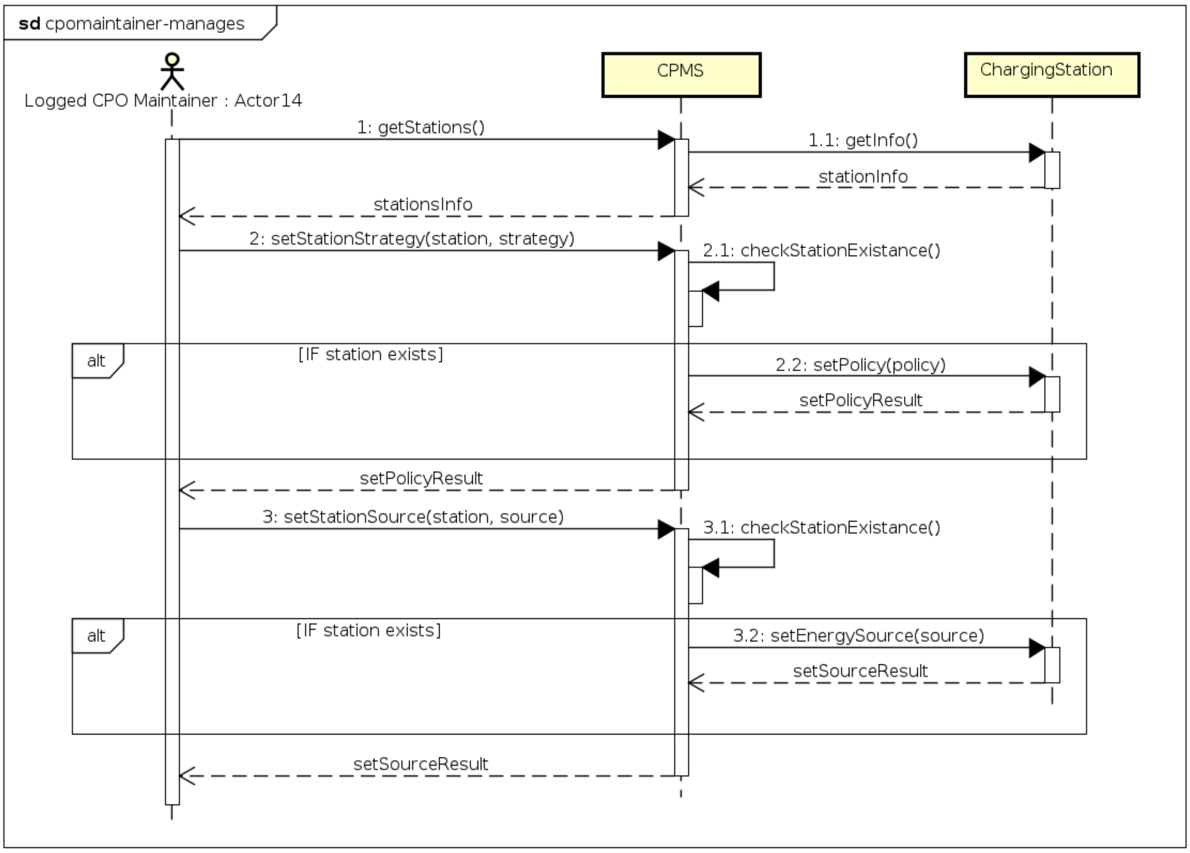


**Figure 18:** CPO maintainer manages a CPMS

## 3.3. Performance requirements

The system in general needs to manage a large collection of electric car users/CPOs and it needs to supply the heaviest services (like computing the cheapest nearest stations) in a reasonable amount of time. Because of that the system shall guarantee a baseline load of 1000000 users/CPOs still with a response time not greater than 5 seconds. To achieve the goal, the system shall be able to decentralize all the computation as possible, trying to make the client responsible of the heaviest loads.

## 3.4. Design constraints

### 3.4.1. Standards compliance

The system must meet the following standards:

- **General Data Protection Regulation (GDPR) law**: The system must be compliant with the current GDPR law about users privacy;
- **Android and iOS**: The system must be compatible with the current versions and reasonably still used previous ones of Android and iOS.

### 3.4.2. Hardware limitations

Because the system consists of a smartphone app, the main hardware limitation is the computational capability of a smartphone processor. Hence the application must be compatible with a low computational capability.

### 3.4.3. Other constraints

TODO MAYBE

## 3.5. Software system attributes

### 3.5.1. Reliability

About the reliability, the system should prefer a fail safe scenarios, where the actual service can behave slower than expected but still consistent with the results. To do so the system should be distributed data wise but also performance wise, allowing a scalability factor while being open for maintenance without completely going down. Some good techniques are Reliable Array of Cloned Services (RACS) and Reliable Array of Partitioned Services (RAPS) which put the reliability very high in the architecture.

### 3.5.2. Availability

Because as stated before a complete period of down would not be great for this type of service, eMall has to prefer the availability over the actual conformity of response time. Thus the availability should be as high as possible but greater than 99.99% and must use some techniques to avoid down time during maintenance.

### 3.5.3. Security

Because the system will handle different personal user data, and because one of the standards that it has to follow is the GDPR law, it is required a certain level of security around the system. So an encryption of the user passwords must be adopted and the access to the user's data must be restricted only to the user itself. It is important that not even the system administrator could access the user's data in respect of the privacy laws.
It needs to be highlighted also that according to the GDPR laws, the user has the right to revoke the consent about the usage of the data by the platform. This means that whenever a user decides to delete the account from the system, all the data must about the user must be deleted permanently.

### 3.5.4. Maintainability

As stated in the Reliability and Availability sections, a good pattern for the whole system would be to consider the maintenance as less invasive as possible, using duplicated data and services. Thus with this idea it would be not complicated to just maintain a single or a restricted amount of nodes per time. This way the user would only experience at worst slowdowns but never actually downtime.

### 3.5.5. Portability

The system should concretize in an APP for the user's smartphone, so it is important to develop the application as cross platform as possible. Doing so eventual updates and modifies won't need any modify to be actually portable from a device to another.

# 4. Formal Analysis Using Alloy

In this section the system described will be modelled and validated using the AlloyTools framework, the analysis is divided in 4 main parts

- Static Analysis
- Dynamic Analysis
- Assertions
- Word Generation

For this analysis the following assumption has been considered

- The Float type (not defined) represent a decimal number
- A CPO can be modelled without being a part of the EMSP

## 4.1. Static Analysis

Here the model is created, all the classes are represented by a sig, for the purpose of this analysis only the relational property are considered, so the attribute of a basic type (such as Int,float,boolean,Data etc. etc.) are not considered. This decision has been taken to simplify the model view and coding, most system that could be used to implement this system already support this type of data, or can be easily implemented and limited in their value ranges.// As a guideline the type are written only in the declarations in a comment, they are defined by abstract type (not implemented) and their ranges are specified; this types are not considered in the rest of the document.

```
module eMall

//-------SIG-----

sig CPO{
cpms: set CPMS}

sig EMSP{
users: set User,
charges: set Charge,
cpos:set CPO
}

sig CPMS{
stations:set ChargingStation,
maintainers:set Maintainer
}

abstract sig Person{
//name: one Str,
//surname: one Str,
  //birthday: one Date,
    //mail: one Str,
  //password: one Str,
}
sig User extends Person{
vehicles: set Vehicle
}
sig Maintainer extends Person{}

sig Vehicle{
//batteryLevel: one Float,
//KWperKm: one Float,
location: one Location,
}{
//inRange[batteryLevel, 0, 100]
//inRange[KWperKm, 0, 100]
}

sig ChargingStation{
position:one Location,
//batteryPresent: one Bool,
//batteryKWh: one Float,
sockets: set ChargingSocket,
strategy: one Strategy
}
{
    //batteryPresent.isTrue implies inRange[batteryKWh, 0, 100]
}

sig ChargingSocket{
chargingType: one ChargingType,
//available: one Bool,
//maximumPowerAmount: one Float,
energySource:one EnergySource
}{
    //inRange[maximumPowerAmount, 0, 100]
}

sig Charge{
//paid: one Bool,
station: one ChargingStation,
user: one User
//amount: one Float,
//date: one Date
}{
    //amount>0
}

abstract sig Strategy{}
one sig Manual extends Strategy{}
one sig Automatic extends Strategy{}

abstract sig ChargingType{}
one sig SuperFast extends ChargingType{}
one sig Fast extends ChargingType{}
one sig Normal extends ChargingType{}

abstract sig EnergySource{}
sig Battery extends EnergySource{
//capacity: one Float
}{
    //capacity>0
}
sig DSO extends EnergySource{}


//utils types

//sig Date{}

//sig Str{}

//simplified using int
sig Location{
// latitude: one Int,
// longitude: one Int
}{//  inRange[latitude, -90, 90] and
// inRange[longitude, -180, 180]
}

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

//fact uniqueMailForUser{
// no disjoint u1,u2: User | u1.mail = u2.mail}

fact uniqueLocationForStation{
no disjoint s1,s2: ChargingStation | s1.position = s2.position}

fact uniqueCPOForCPMS{
no disjoint c1,c2: CPO, cp:CPMS | cp in c1.cpms and cp in c2.cpms}
```

```
fact uniqueStationForCPMS{
no disjoint c1,c2: CPMS, s:ChargingStation | s in c1.stations and s in c2.stations}

fact socketOnlyOneStation{
all s:ChargingSocket| s in ChargingStation.sockets
no disjoint c1,c2: ChargingStation,
s:ChargingSocket|(s in c1.sockets and s in c2.sockets)}

fact noVehicleWithoutUser{
all v:Vehicle|  v in User.vehicles}

fact noStationWithoutCPMS{
all s:ChargingStation|  s in CPMS.stations}

fact noUserWithoutEMSP{
all u:User|  u in EMSP.users}

fact noChargeWithoutEMSP{
all c:Charge|  c in EMSP.charges}

fact noChargeWithoutUserInTheEMSP{
all c:Charge| c in EMSP.charges and c.user in EMSP.users
}

fact allChargeAreFromChargingStationInTheSystem{
all s:Charge.station | s in EMSP.cpos.cpms.stations
}
fact maintainersMantainStationOfTheSameCPO{
all m:Maintainer, c1,c2:CPO|(not c1=c2 and m in c1.cpms.maintainers)
implies m not in c2.cpms.maintainers
}
```

## 4.2. Dynamic Programming

In this part the major operation are described and run, as a convection letter1 represent the old version (before the execution), while letter0 is the new version after the execution of the predicate.

### 4.2.1. User books a charge

```
    pred UserCreatesACharge(e,e1:EMSP,u:User, s:ChargingStation){
        one c:Charge  | u in e1.users and
        c.user=u and c.station=s and  (not (e1 = e)) and  e1.users=e.users and
        e1.cpos=e.cpos and e1.charges=e.charges+c
        }
        run UserCreatesACharge for 3 but exactly 2 EMSP
```

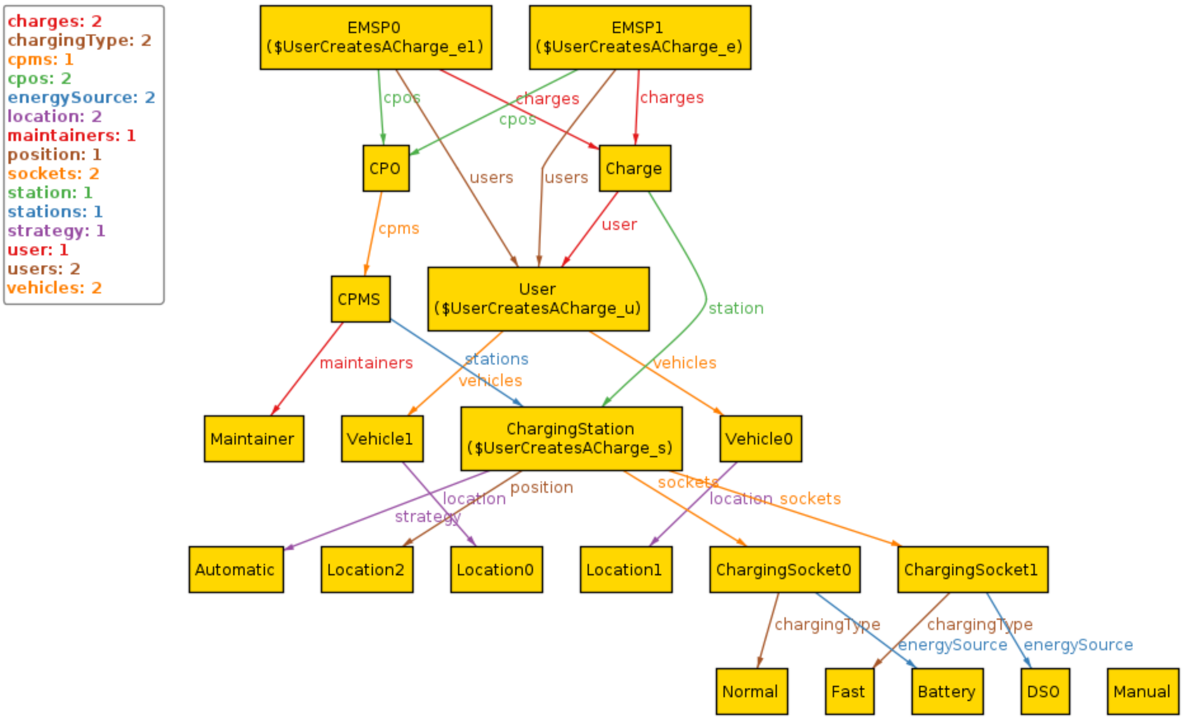


**Figure 19:** Added Charge

### 4.2.2. CPO subscribe to EMSP

```
pred CPOSubscribeItselfToEMSP(e,e1:EMSP,cpo:CPO){
     not (e1 = e)
    e.charges=e1.charges
    e.users= e1.users
    e.cpos=e1.cpos+cpo
    }
    run CPOSubscribeItselfToEMSP for 3 but exactly 2 EMSP, exactly 2 CPO
```

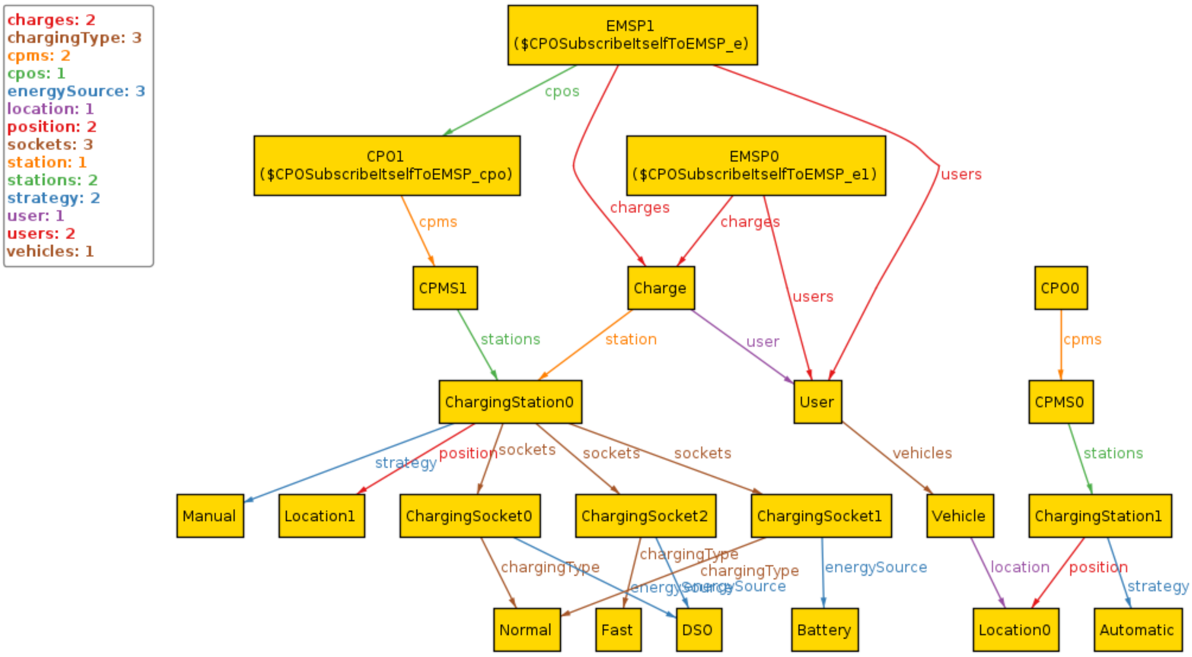


**Figure 20:** CPO subscribed

### 4.2.3. CPO add CPMS

```
pred CPOAddCPMS(c,c1:CPO,cp:CPMS){
     not (c1 = c)
   c.cpms=c1.cpms+cp
   }
   run CPOAddCPMS for 3 but exactly 2 CPO, exactly 2 CPMS
```

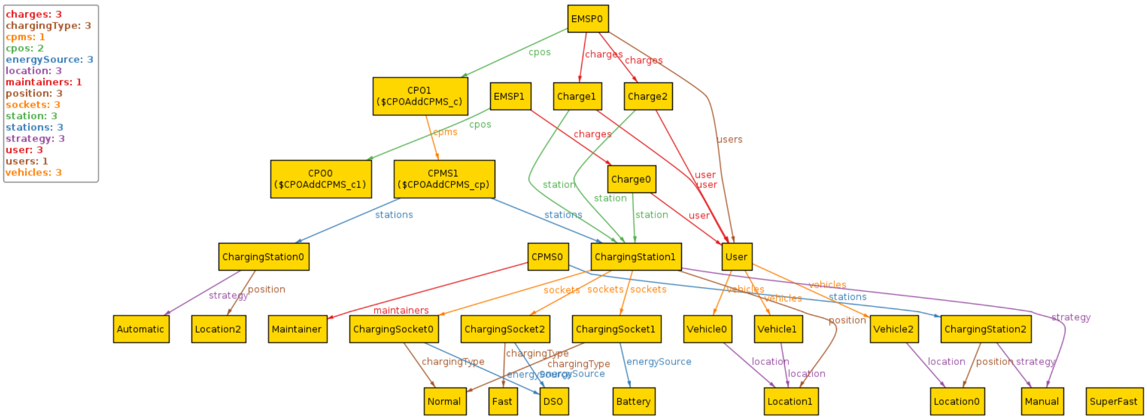


**Figure 21:** Added CPMS

### 4.2.4. CPO add mantainer to CPMS

```
pred CPOAddMantainerToCPMS(c:CPO,cp,cp1:CPMS,m:Maintainer){
     not (cp = cp1)
     cp1 in c.cpms
     cp in c.cpms
     cp.stations=cp1.stations
     cp.maintainers=cp1.maintainers+m
    }
    run CPOAddMantainerToCPMS for 3 but exactly 2 CPMS
```

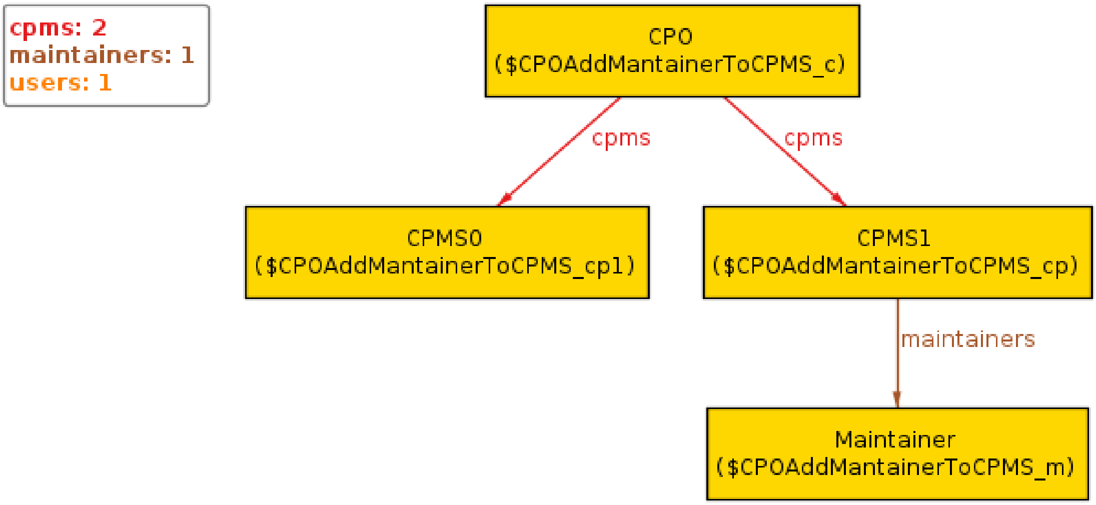

cpms: 2
maintainers: 1
users: 1
CPO
($CPOAddMantainerToCPMS_c)
cpms
cpms
CPMS0
($CPOAddMantainerToCPMS_cp1)
CPMS1
($CPOAddMantainerToCPMS_cp)
maintainers
Maintainer
($CPOAddMantainerToCPMS_m)

**Figure 22:** Added Maintainer

### 4.2.5. CPO add station to CPMS

```
pred CPOAddStationToCPMS(c:CPO,cp,cp1:CPMS,s:ChargingStation){
     not (cp = cp1)
     cp1 in c.cpms
     cp in c.cpms
     cp.maintainers = cp1.maintainers
     cp.stations=cp1.stations+s
    }
    run CPOAddStationToCPMS for 3 but exactly 2 CPO
```

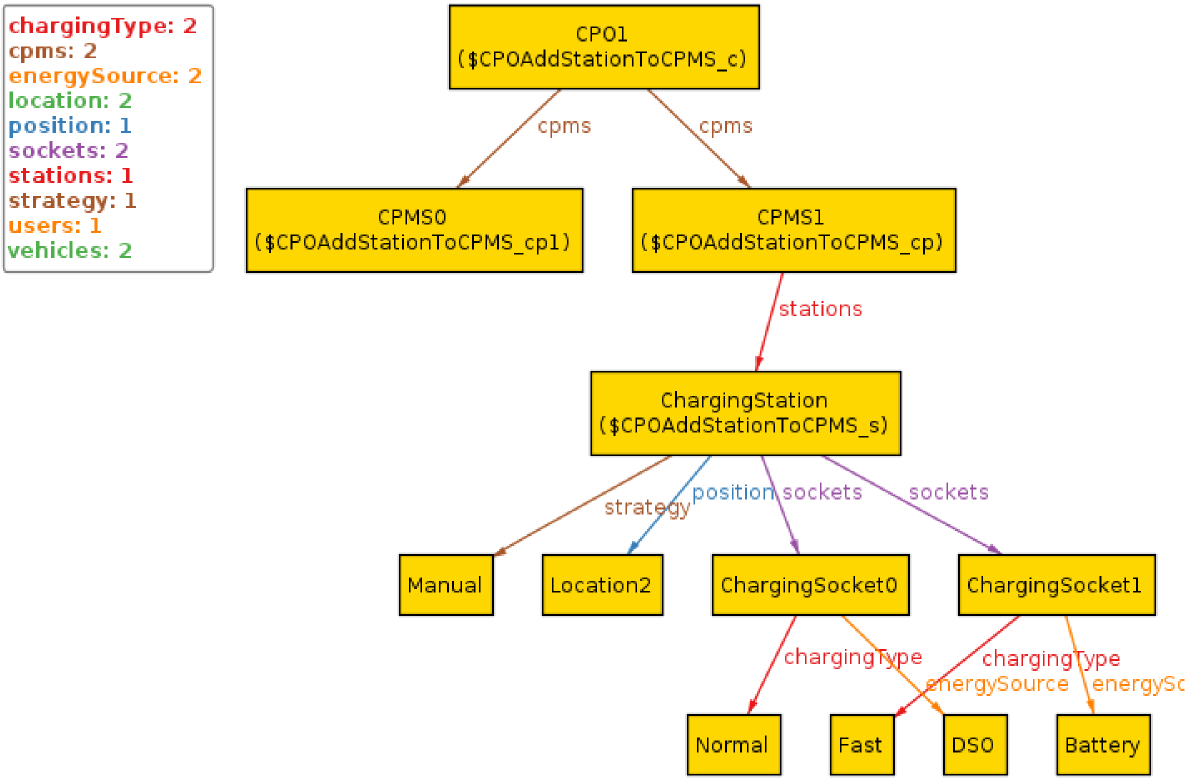


**Figure 23:** Added Station

### 4.2.6. CPO add socket to station

The following code does not solve, for a limitation of the language,because as a fact a socket can exist only in one station; this is due of the Add patter (s=new station, s1=old station) as for now this pred is never run but supposed as consistent

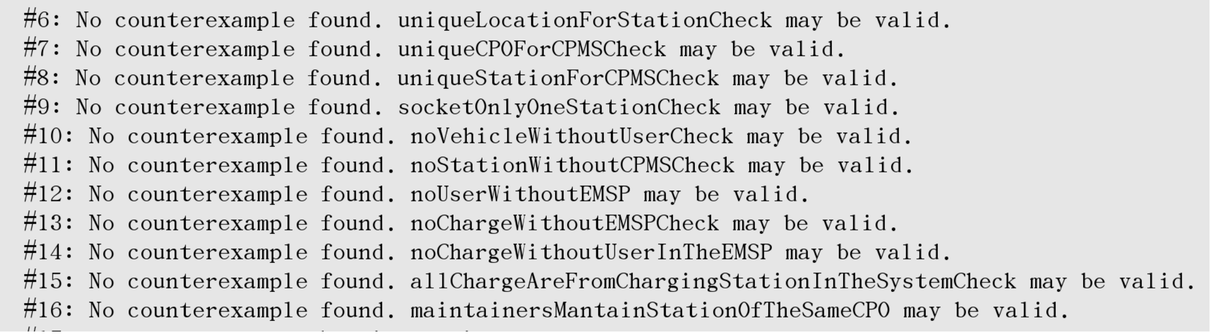
```
pred CPOAddSocketToStation(c:CPO,cp:CPMS, s,s1:ChargingStation,sk:ChargingSocket){
    not (s = s1)
     cp in c.cpms
     s in cp.stations
     s1 in cp.stations
     s.position=s1.position
     s.strategy=s1.strategy
     s1.sockets=s.sockets+sk
   }
   run CPOAddSocketToStation for 3 but exactly 2 ChargingStation
```

### 4.3. Assertions

Here we check the validity of the model trough the Assert notation.

```
assert uniqueLocationForStationCheck{
  no disjoint s1,s2: ChargingStation | s1.position = s2.position}
check uniqueLocationForStationCheck for 10

assert uniqueCPOForCPMSCheck{
no disjoint c1,c2: CPO, cp:CPMS | cp in c1.cpms and cp in c2.cpms}
check uniqueCPOForCPMSCheck  for 10

assert uniqueStationForCPMSCheck{
no disjoint c1,c2: CPMS, s:ChargingStation | s in c1.stations and s in c2.stations}
check uniqueStationForCPMSCheck for 10

assert socketOnlyOneStationCheck{
   all s:ChargingSocket| s in ChargingStation.sockets
no disjoint c1,c2: ChargingStation, s:ChargingSocket|(s in c1.sockets and s in c2.socket
check socketOnlyOneStationCheck for 10

assert noVehicleWithoutUserCheck{
all v:Vehicle|  v in User.vehicles}
check noVehicleWithoutUserCheck for 10

assert noStationWithoutCPMSCheck{
all s:ChargingStation|  s in CPMS.stations}
check noStationWithoutCPMSCheck for 10

assert noUserWithoutEMSP{
all u:User|  u in EMSP.users}
check noUserWithoutEMSP for 10

assert noChargeWithoutEMSPCheck{
all c:Charge|  c in EMSP.charges}
check noChargeWithoutEMSPCheck for 10

assert noChargeWithoutUserInTheEMSP{
all c:Charge| c in EMSP.charges and c.user in EMSP.users}
check noChargeWithoutUserInTheEMSP for 10

assert allChargeAreFromChargingStationInTheSystemCheck{
all s:Charge.station | s in EMSP.cpos.cpms.stations }
check allChargeAreFromChargingStationInTheSystemCheck for 10

assert maintainersMantainStationOfTheSameCPO{
all m:Maintainer, c1,c2:CPO|(not c1=c2 and m in c1.cpms.maintainers) implies m not in c2
check maintainersMantainStationOfTheSameCPO for 10
```

Which generate the following output.



```
#6: No counterexample found. uniqueLocationForStationCheck may be valid.
#7: No counterexample found. uniqueCPOForCPMSCheck may be valid.
#8: No counterexample found. uniqueStationForCPMSCheck may be valid.
#9: No counterexample found. socketOnlyOneStationCheck may be valid.
#10: No counterexample found. noVehicleWithoutUserCheck may be valid.
#11: No counterexample found. noStationWithoutCPMSCheck may be valid.
#12: No counterexample found. noUserWithoutEMSP may be valid.
#13: No counterexample found. noChargeWithoutEMSPCheck may be valid.
#14: No counterexample found. noChargeWithoutUserInTheEMSP may be valid.
#15: No counterexample found. allChargeAreFromChargingStationInTheSystemCheck may be valid.
#16: No counterexample found. maintainersMantainStationOfTheSameCPO may be valid.
```

**Figure 24:** Assertion output

## 4.4. Word Generation

Here is the code of the word generation.

```
pred show() {
#EMSP = 1
#CPO>2
#Charge>2
#Vehicle>2
#User>2
}
run show
```

And the generated word.

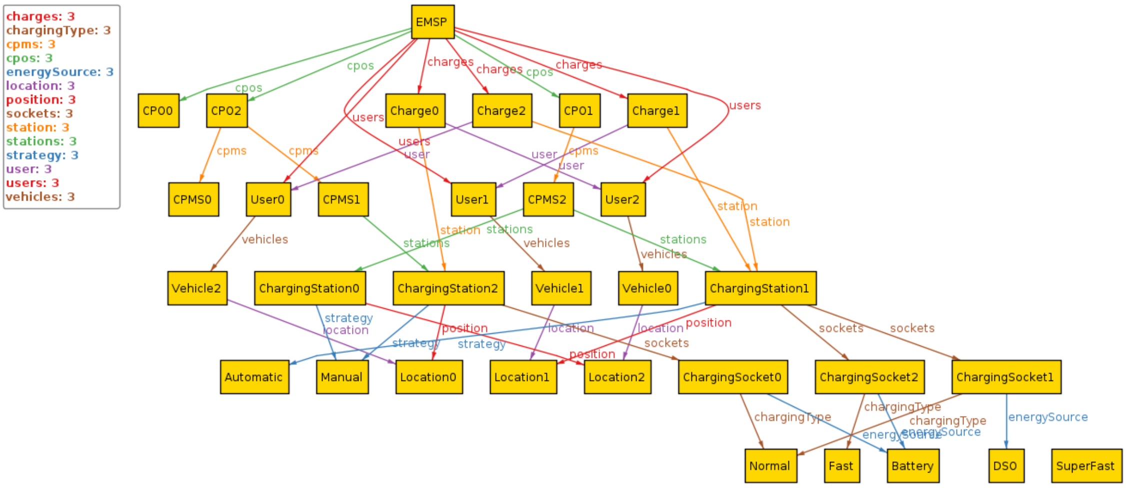


**Figure 25:** Generated Word

## 5. Effort Spent

- 15/11/2022: 15:00 - 18:00 Federico, Emilio, Matteo
- 16/11/2022: 08:30 - 10:00 Emilio
- 17/11/2022: 21:00 - 23:00 Federico, Emilio, Matteo
- 18/11/2022: 10:00 - 12:00 Emilio, Federico
- 21/11/2022: 19:00 - 20:00 Matteo
- 22/11/2022: 14:30 - 16:00 Matteo
- 23/11/2022: 10:30 - 11:30 Matteo
- 24/11/2022: 21:30 - 22:30 Matteo, Federico
- 25/11/2022: 09:00 - 09:30 Federico
- 25/11/2022: 19:00 - 19:30 Matteo
- 26/11/2022: 08:30 - 09:00 Federico
- 26/11/2022: 16:00 - 17:00 Federico, Emilio, Matteo
- 28/11/2022: 08:30 - 09:00 Federico
- 28/11/2022: 10:00 - 12:00 Emilio
- 30/11/2022: 22:00 - 23:00 Emilio
- 28/11/2022: 08:00 - 08:30 Federico
- 01/12/2022: 16:00 - 17:30 Matteo
- 01/12/2022: 20:30 - 21:30 Emilio
- 01/12/2022: 21:30 - 23:00 Federico, Emilio, Matteo
- 04/12/2022: 19:00 - 20:00 Emilio
- 05/12/2022: 09:00 - 09:30 Federico
- 05/12/2022: 11:00 - 11:45 Emilio
- 05/12/2022: 15:00 - 16:30 Matteo
- 05/12/2022: 19:15 - 19:50 Emilio
- 06/12/2022: 15:30 - 17:00 Emilio, Matteo
- 07/12/2022: 14:00 - 15:00 Matteo
- 10/12/2022: 20:00 - 20:30 Matteo
- 11/12/2022: 10:30 - 12:00 Federico
- 11/12/2022: 15:10 - 16:40 Matteo
- 12/12/2022: 10:00 - 12:00 Emilio

- 12/12/2022: 10:30 - 12:00 Emilio
- 12/12/2022: 12:30 - 13:00 Matteo
- 12/12/2022: 15:00 - 16:30 Federico, Emilio, Matteo
- 12/12/2022: 17:30 - 18:30 Emilio
- 12/12/2022: 19:00 - 19:30 Matteo
- 12/12/2022: 22:00 - 23:00 Federico
- 13/12/2022: 15:15 - 17:00 Emilio, Matteo
- 15/12/2022: 10:00 - 16:00 Federico
- 17/12/2022: 17:00 - 01:00 Federico
- 17/12/2022: 10:30 - 12:00 Federico, Emilio, Matteo
- 17/12/2022: 21:00 - 22:00 Federico, Emilio, Matteo
- 18/12/2022: 09:30 - 11:30 Matteo
- 18/12/2022: 09:30 - 12:00 Federico
- 18/12/2022: 16:30 - 21:00 Federico
- 20/12/2022: 09:00 - 12:30 Federico
- 19/12/2022: 09:30 - 11:30 Emilio
- 20/12/2022: 14:00 - 15:30 Emilio
- 21/12/2022: 10:30 - 11:45 Matteo
- 21/12/2022: 10:45 - 12:00 Emilio
- 21/12/2022: 12:00 - 22:00 Federico
- 21/12/2022: 14:00 - 15:40 Emilio
- 21/12/2022: 17:15 - 18:15 Matteo
- 21/12/2022: 17:20 - 18:10 Emilio

## 6. References

- **RACS** and **RAPS**: `https://arxiv.org/pdf/cs/9912010.pdf`
- **Partita IVA**: `https://www.agenziaentrate.gov.it/portale/web/guest/agenzia/amministrazi` `servizi-erogati/carta-servizi/i-nostri-servizi/area-identificazione-del-contribuent` `partita-iva`