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

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.
e-Mobility for All (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, their environmental friendliness and any special offer that they might have. It will allow the users to book, cancel and pay for a charge and it will notify them 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 Charge Point Management System (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, cancel and pay for a charge;
G3 The eMSP shall allow the user to perform a charge;
G4 CPMSs shall handle the vehicle charging cycles;
G5 CPMSs shall manage the charging stations;

1.2. Scope

WP1 People charge electric vehicles in different modes (NORMAL, FAST, SUPER-FAST);
WP2 People use web calendar;
WP3 People pay for their charging service;
WP4 DSOs supply energy to CPOs;
WP5 Some CPOs own batteries;
WP6 CPOs have Partita IVA;
WP7 CPOs decide whether to use batteries or DSO supplied energy;
SP1 The eMSP suggests the user to charge the vehicle;
SP2 The eMSP notifies the user when the charging process is finished;
SP3 CPMSs acquire information about energy prizes from DSOs;
SP4 The user books a charge using the eMSP;
SP5 The user asks the eMSP for suggestions about charging station;
SP6 The user pays for the service using the eMSP;
SP7 CPOs gather the energy source through the CPMS;

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. 1, 5, 8, 10, 11, 12, 15, 48

1.3.2. Acronyms

eMall	e-Mobility for All
eMSP	e-Mobility Service Provider
CPO	Charging Point Operator
CPMS	Charge Point Management System
DSO	Distribution System Operator
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. 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. Formal descriptions about the world (world phenomena) and the interactions between the system and the world (shared phenomena) are provided. At the end of the introduction there is a reference subsection for definitions and revision history;
- **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 cases 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 of the problem in

Alloy language, 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 reference 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 **S12** and **S14** we assume that the CPOmaintainer is already logged in the CPMS (**S11**).

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); 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 and 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, environmental friendliness 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 and the charging speed. 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 designed 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 CPO inserts the revenue percentage:
Andy, the CEO of a CPO, decides to set a different revenue percentage. He opens the eMall website, logs in and inserts the wanted revenue percentage;

S11 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;

S12 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;

S13 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;

S14 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

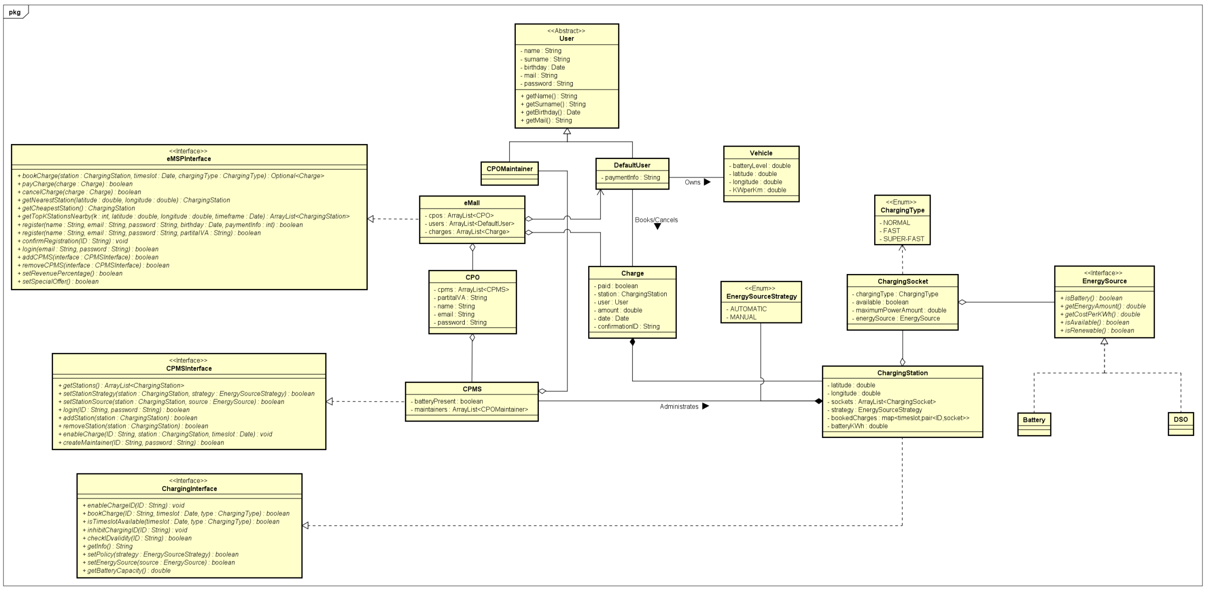


Figure 1: UML

2.1.3. State diagrams

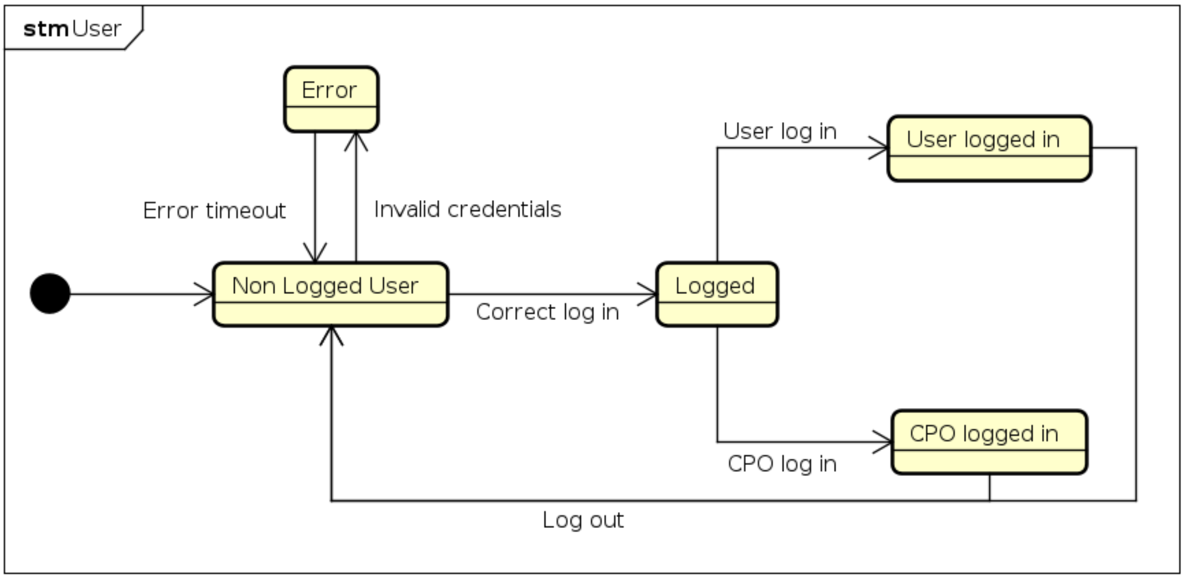


Figure 2: User state diagram

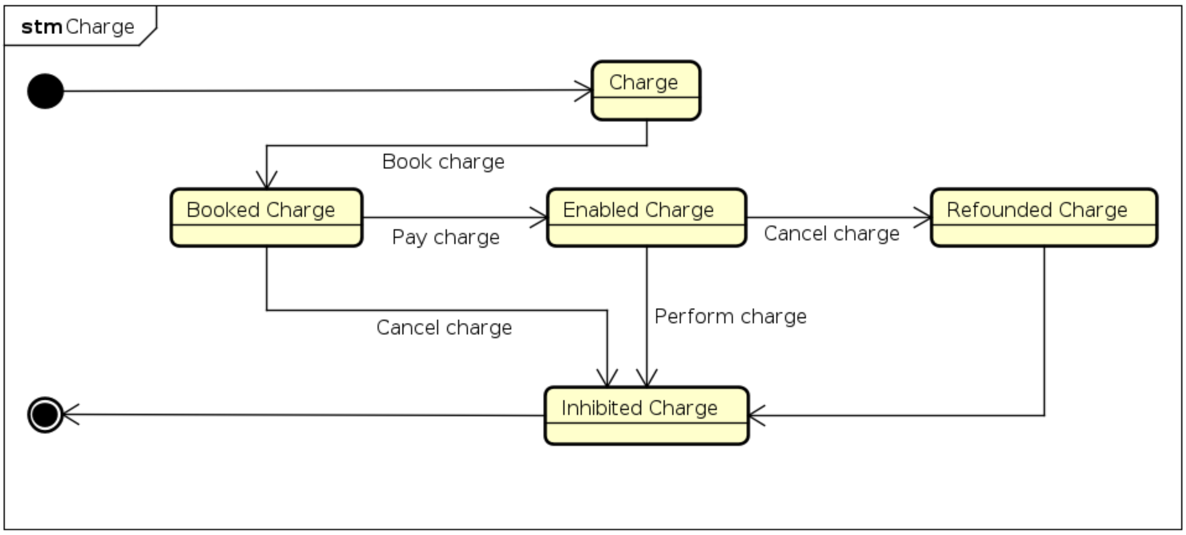


Figure 3: Charge state diagram

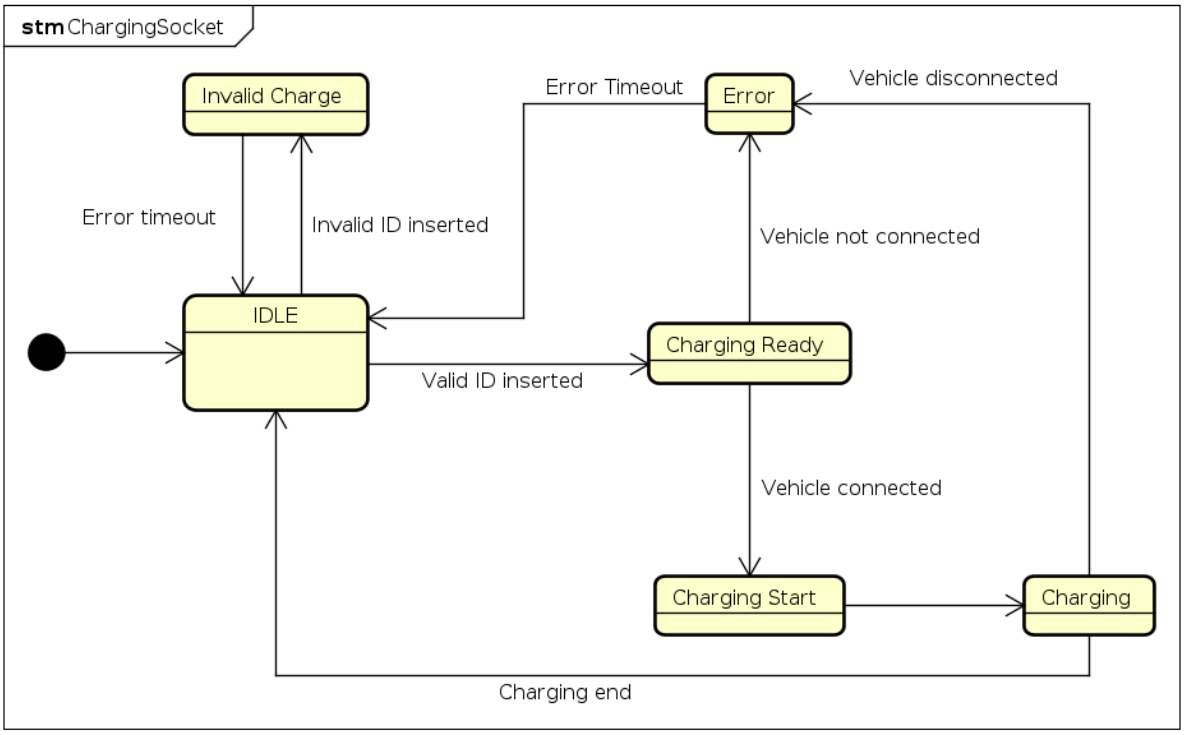


Figure 4: Charging socket state diagram

2.2. Product functions

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

2.2.1. e-Mobility for All (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. Charge Point Management System (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 revenue percentage 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 the eMall services for users;
- **A2 Registered user:** A user that has access to all the eMall features. This actor can be 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 **Unregistered CPO** A CPO that needs to register before accessing the eMall services for CPOs;

A4 **Registered CPO** A CPO that can add/remove to eMall CPMSs;

A5 **Registered CPOmaintainer:** A user that has access to all the CPMS features. These CPMS allows the maintainer to configure the energy source strategy, add other CPO maintainers and to visualize all the charging stations statuses;

2.4. Assumptions 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 users have an active internet and GPS connection always available while using the service;

DA5 Once installed, the initial login to a CPMS is done using the default user and password and the first CPOmaintainer is configured;

DA6 Charging sockets have internet connection and an appropriate interface;

DA7 Charging sockets have an input method for inserting the pin so that the user can validate the charge;

DA8 The software utilizes payment APIs;

2.4.2. Constraint

C1 If a User wants to use a charging station he must have booked and paid a charge through the system;

C2 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 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 has to input the email and password in order to authenticate. From the *login* page there is also the possibility to go to the *sign up* page where the fields for inserting the necessary information are shown. After logging in, there are multiple tabs in the app, which are:

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

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

A CPO can register to the system with a special form providing Company name, email, password and Partita IVA. Once the CPO is registered and logged in, he can insert/remove the API reference to the interface of a CPMS.

CPMS The CPMS works as a web application; When a CPOmaintainer logs in he has the possibility to handle all the added stations.
In particular he can view: the system status, the list of charging stations and their policy, the available sockets and the State of Charge (SoC) of batteries.
The CPOmaintainers can change the policy of each charging station; in particular they can:

- Choose a the energy provider from a list of DSOs;
- Choose to use only the energy stored in the batteries and, once discharged, to go into automatic mode;
- Choose the automatic mode, that will pick autonomously which DSO to use, whether 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. Finally, the CPO can also create some special offers.

3.1.2. Hardware interfaces

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).

3.1.3. Software interfaces

eMSP In order to provide all the functionalities, the eMSP should provide the following software interfaces:

- Register to the eMSP as a user providing name, email, password, birthday, payment info;
- Register to the eMSP as a CPO providing name, email, password, Partita IVA;
- Confirm the registration of a given user;
- Login to the eMSP providing correct email and password;
- Retrieving data about charging stations, such as the nearest (providing a location), the cheapest or a list of K charging stations;
- Book a charge providing charging station, time slot and type of charge;
- Cancel an already booked charge;
- Paying a charge with the already set payment method;
- A CPO can add a CPMS providing a link to its API interface;

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

- Login to the CPMS providing an ID and a password;
- Enable a charge on a charging station, passing the ID (a PIN in order to authorize the charge once the user gets in the station), the station and the time slot;
- 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;

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 manageable 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 external functions made by users (like booking a charge, canceling a charge, seeing what time slots 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 information about a selected station such as types of available sockets, price for the charge, location, available time slots, parameter on environmental friendliness;

R4 The eMSP shall reserve a socket in the right charging station for a user who booked 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 shall occur;

R6 The eMSP shall allow the user to pay for a booked charge;

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 web calendar, retrieve information about the appointments and parse them;

R10 The eMSP shall be able to use the information 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;

R12 The eMSP shall aggregate 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 its API reference;

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

R16 The eMSP shall allow the CPO to set the wanted revenue percentage;

R17 The eMSP shall allow the CPO to set special offers;

R18 The CPMS shall be reachable by eMSPs in order to perform or cancel a booking, or query the system;

R19 The CPMS shall allow the CPOmaintainer to access to the system;

R20 The CPMS shall allow the CPOmaintainer to modify the information about their systems, such as adding/removing charging stations, set stations sources and create/remove maintainers;

R21 The CPMS shall allow the CPOmaintainer to choose the energy source and strategy;

R22 The CPMS shall allow the CPOmaintainer to choose manual or automatic mode;

[1] This parameter may be set by the user

Requirements/Goals:	G1	G2	G3	G4	G5
R1	X	X	X		
R2	X	X	X		
R3	X		X		
R4		X	X		
R5		X	X		
R6		X			
R7		X			
R8	X	X			
R9	X	X			
R10	X	X			
R11			X		
R12		X			
R13		X		X	
R14		X		X	
R15		X		X	
R16					X
R17					X
R18	X	X	X	X	
R19				X	X
R20					X
R21				X	X
R22				X	X

Table 1: Linking table among goals and requirements

3.2.1. Use cases

Actor:	User
Entry conditions:	The user doesn't have any account on the platform.
Event flow:	1. The user presses the *Register* button; 2. The user enters name, surname, birthday, email, password, payment method; 3. The user presses the confirmation button; 4. The system sends a confirmation email to the provided email address; 5. The user presses a link in the confirmation email; 6. The system shows a success message.
Exit condition:	The user now has an active account.
Exceptions:	• The user takes more than 15 minutes to press the confirmation email: In this case the account won't be created, the system will notify the user and all the process should be redone; • The user inserts an already registered email to the system: the system notifies the user without letting the process continue; • The user doesn't insert some required information: the system notifies the user without letting the process continue.

Table 2: Unregistered user

Other use cases to do:
- CPO sets a special offer
- user performs a charge
- system suggests charge via calendar
- CPOmaintainer logs into CPMS
- CPOmaintainer adds stations to CPMS
- CPOmaintainer manages CPMS

Actor:	CPO
Entry conditions:	The CPO isn't registered yet.
Event flow:	1. The CPO selects the *Register as CPO* option; 2. The CPO enters company name, email, password and Partita IVA; 3. The system sends a confirmation email to the provided email address; 4. The CPO presses a link in the confirmation email; 5. The system shows a success message.
Exit condition:	The CPO is now registered to the platform.
Exceptions:	• The CPO takes more than 15 minutes to press the confirmation email: In this case the account won't be created, the system will notify the CPO and all the process should be redone; • The CPO inserts an already registered email to the system: the system notifies the CPO without letting the process continue; • The CPO doesn't insert some required information: the system notifies the CPO without letting the process continue; • The Partita IVA isn't valid or can't be verified: the system notifies the CPO without letting the process continue.

Table 3: Unregistered CPO

Actor:	User or CPO
Entry conditions:	The actor is not yet logged in.
Event flow:	1. The actor presses the login button; 2. The actor inserts email and password; 3. The system validates the credentials; 4. The system lets the actor access the available features.
Exit condition:	The actor is logged into the system and can perform some specific actions (i.e. for the user: search near stations, book a charge, cancel a charge, pay for a charge and view previously booked charges; for the CPO: add a CPMS, remove a CPMS, set revenue percentage, set special offers).
Exceptions:	• The user inserts wrong credentials: The system notifies it with an error message.

Table 4: User or CPO logs in

Actor:	CPO
Entry conditions:	The CPO is already logged in the system.
Event flow:	1. The CPO selects the option to add a CPMS; 2. The CPO inserts the API reference in the form and sends it; 3. The system checks if it exists and if it is reachable; 4. The system adds the CPMS to the available CPMSs and retrieves all the charging stations attached to the CPMS.
Exit condition:	A CPMS is added to the system and can be reached to book charges or other queries by users.
Exceptions:	• The API reference doesn't exist, the CPMS isn't reachable or the CPMS was already added to the system previously: the system returns an error message with the specific reason.

Table 5: CPO adds a CPMS to the system

Actor:	CPO
Entry conditions:	The CPO is already logged into the system.
Event flow:	1. The CPO selects the option to insert the wanted revenue percentage for a charge performed on a charging station owned by the CPO; 2. The CPO inputs the wanted revenue percentage; 3. The system checks the inputted value; 4. The system shows a successful message.
Exit condition:	The revenue percentage is changed to the new inputted value.
Exceptions:	• The value inserted is not a positive value (i.e. the value is below 0%): the system returns an error message.

Table 6: A CPO sets the revenue percentage

Actor:	User
Entry conditions:	The user is already logged into the system.
Event flow:	1. The user presses the *Map* button; 2. The user inserts the location and time frame; 3. The user presses the *Search* button; 4. The system shows a ordered list of k best nearby stations.
Exit condition:	The user has a list of ordered nearby stations.
Exceptions:	• The inserted time frame isn't correct: the system returns an error message.

Table 7: The user searches for nearby stations

Actor:	User
Entry conditions:	The user is already logged into the system and he has already searched for the station.
Event flow:	1. The user inserts the time frame; 2. The user presses the *Book* button; 3. The system shows the charge ID and sends the user an email with the details.
Exit condition:	The user has now a booked charge.
Exceptions:	• The inserted time frame isn't correct: the system returns an error message; • The inserted time frame is not available: the system returns an error message.

Table 8: User books a charge

Actor:	User
Entry conditions:	The user is already logged into the system and there is an already existing booked charge.
Event flow:	1. The user presses the *Pay* button near the booked charge; 2. The system asks for a confirmation; 3. The user presses the *Confirm* button; 4. The system performs the payment, marks the charge as enabled and sends a confirmation email.
Exit condition:	A booked charge is paid.
Exceptions:	• The charge does not exist: the system returns an error message; • The charge does not belong to the user: the system returns an error message; • The payment fails: the system returns an error message.

Table 9: User pays for a charge

Actor:	User
Entry conditions:	The user is already logged into the system and there is an already existing booked charge.
Event flow:	1. The user presses the *Cancel* button near the booked charge; 2. The system asks for a confirmation; 3. The user presses the *Confirm* button; 4. The system cancels the charge and in case of an already paid one, it refunds the user; 5. The system sends a confirmation email.
Exit condition:	A previously booked charge is canceled.
Exceptions:	• The charge does not exist: the system returns an error message; • The charge does not belong to the user: the system returns an error message.

Table 10: User cancels a charge

3.2.2. Use case diagrams

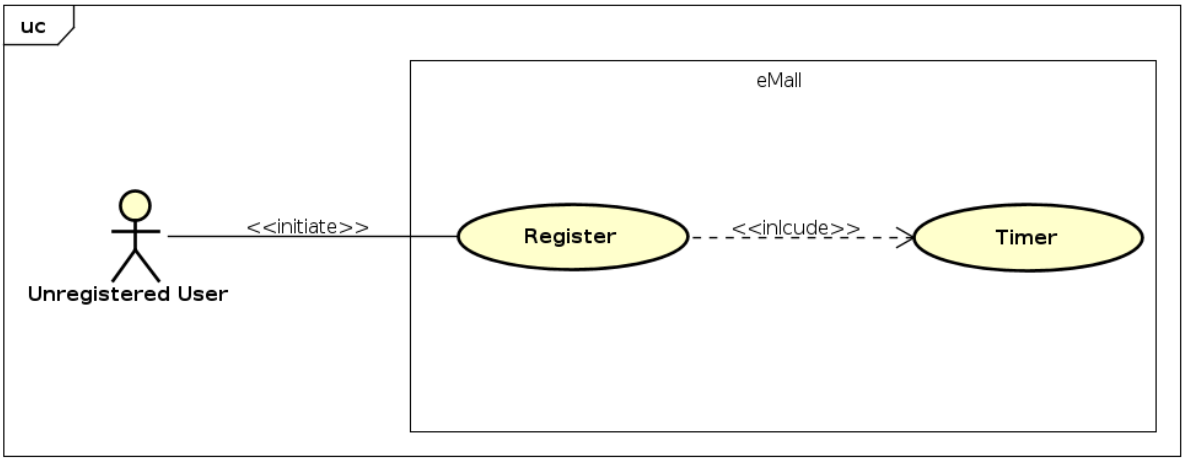


Figure 5: Unregistered user use case

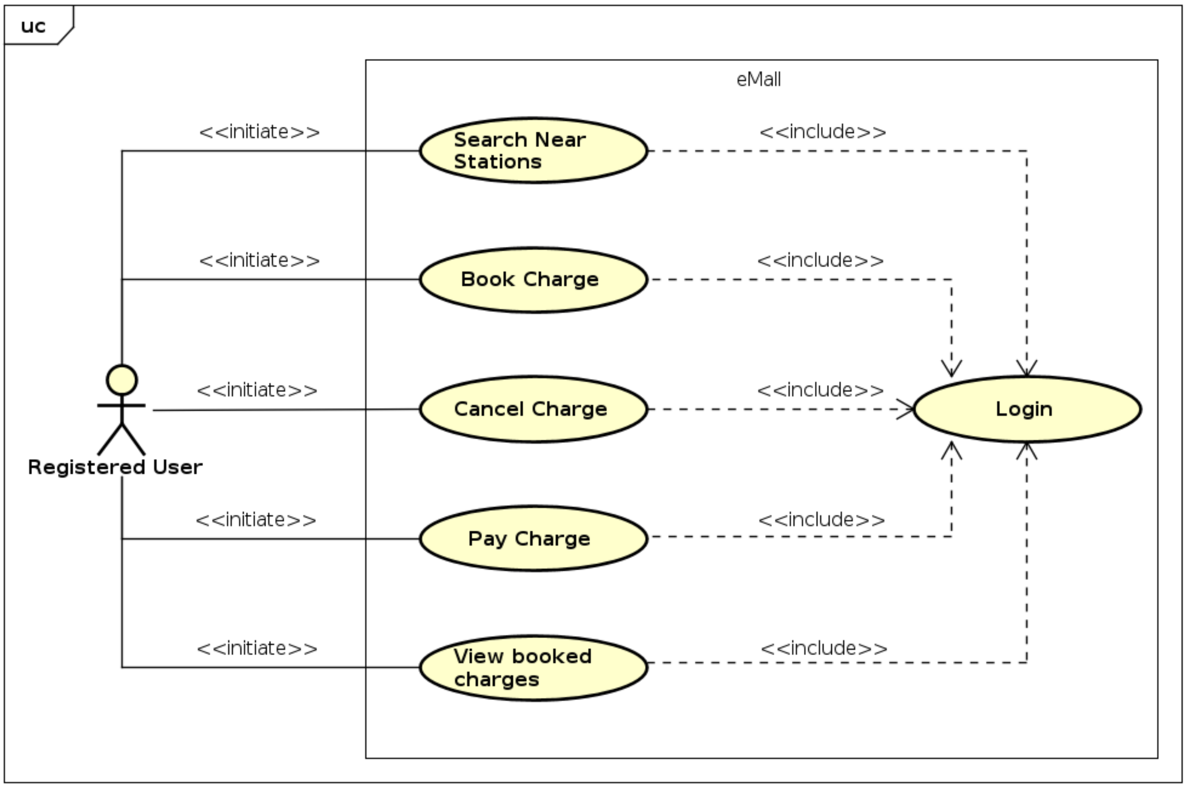
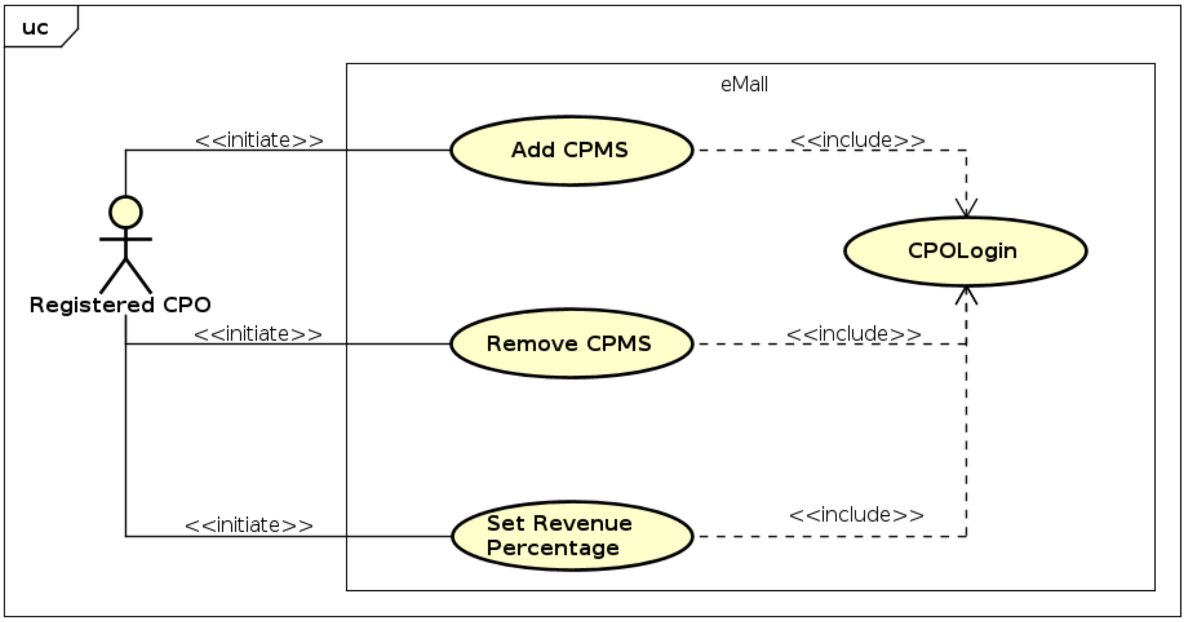


Figure 6: Registered user use case



Adding set special offers in "Registered CPO" use case diagram

Figure 7: Registered CPO

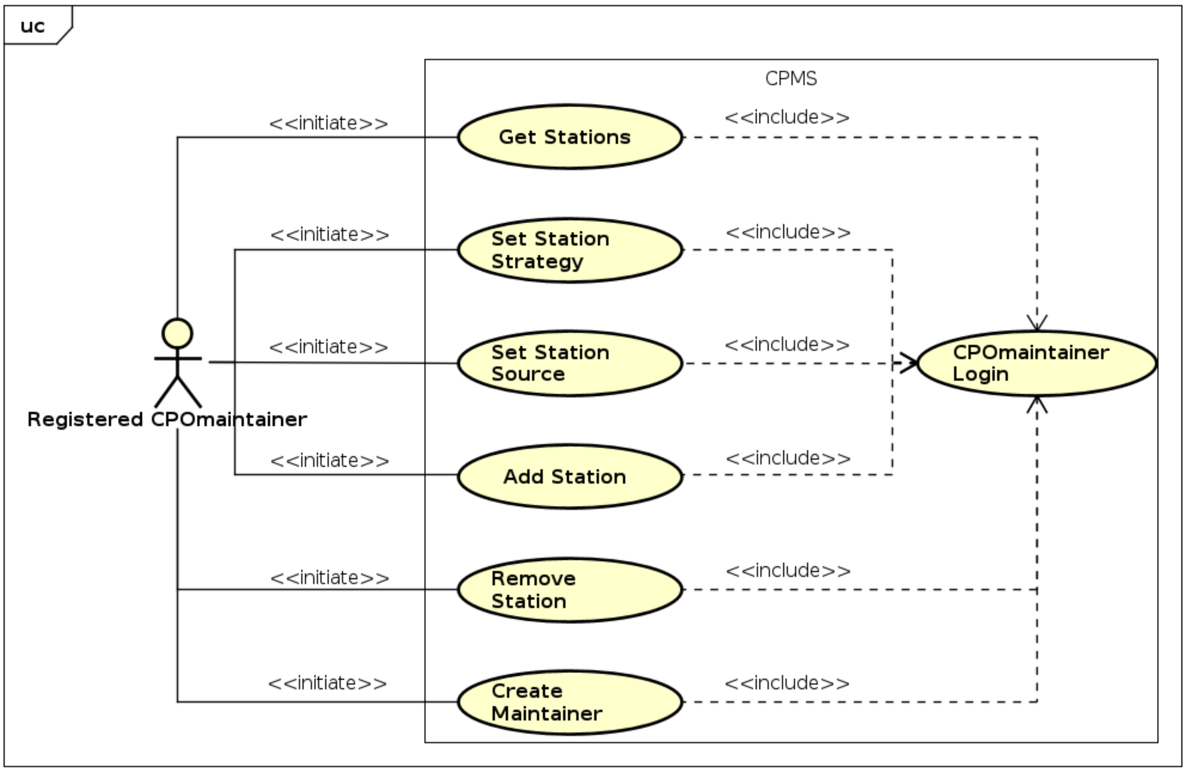


Figure 8: Registered CPOmaintainer

3.2.3. Sequence diagrams

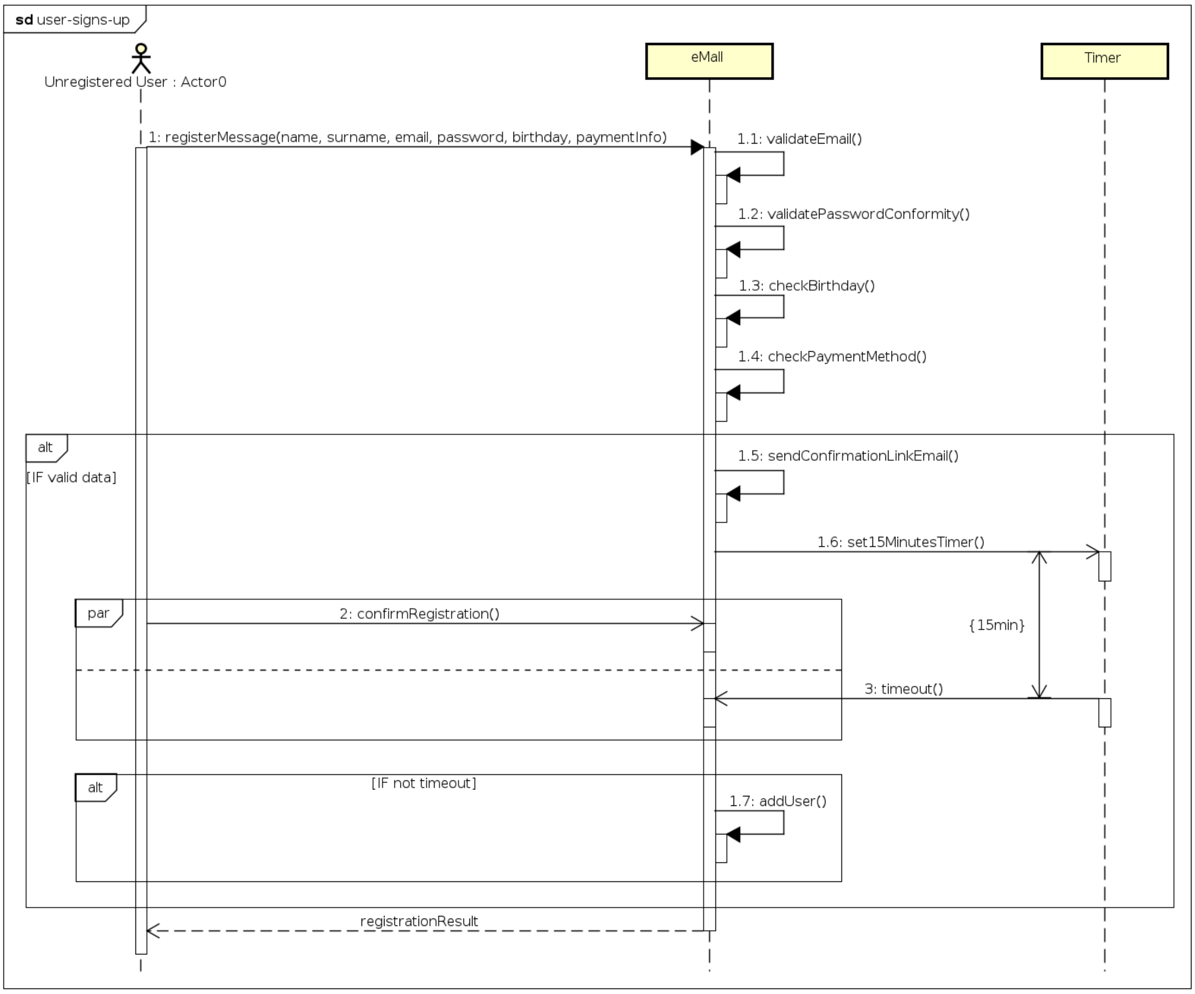


Figure 9: Registration into eMall sequence

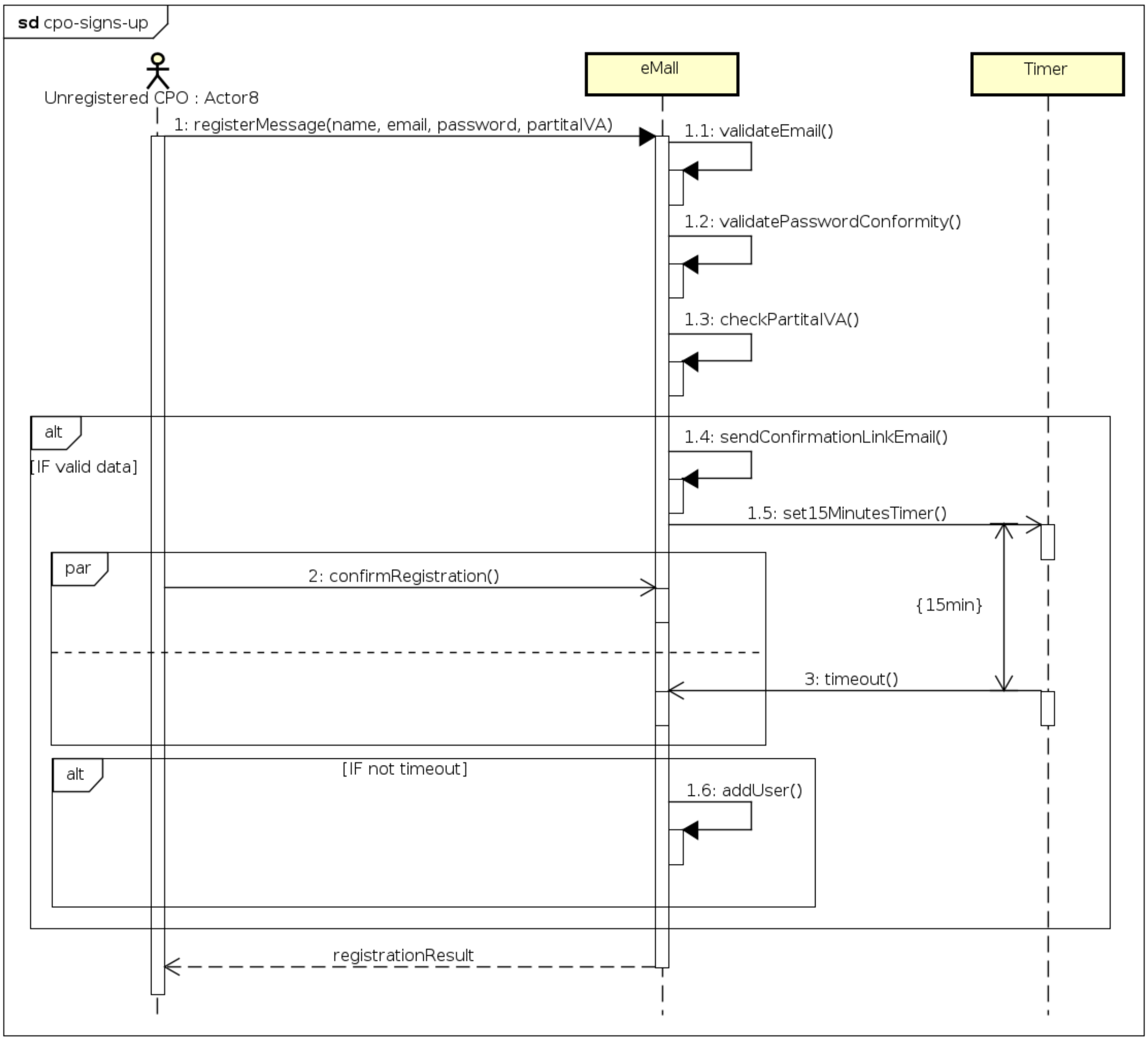


Figure 10: Registration of CPO into eMall sequence

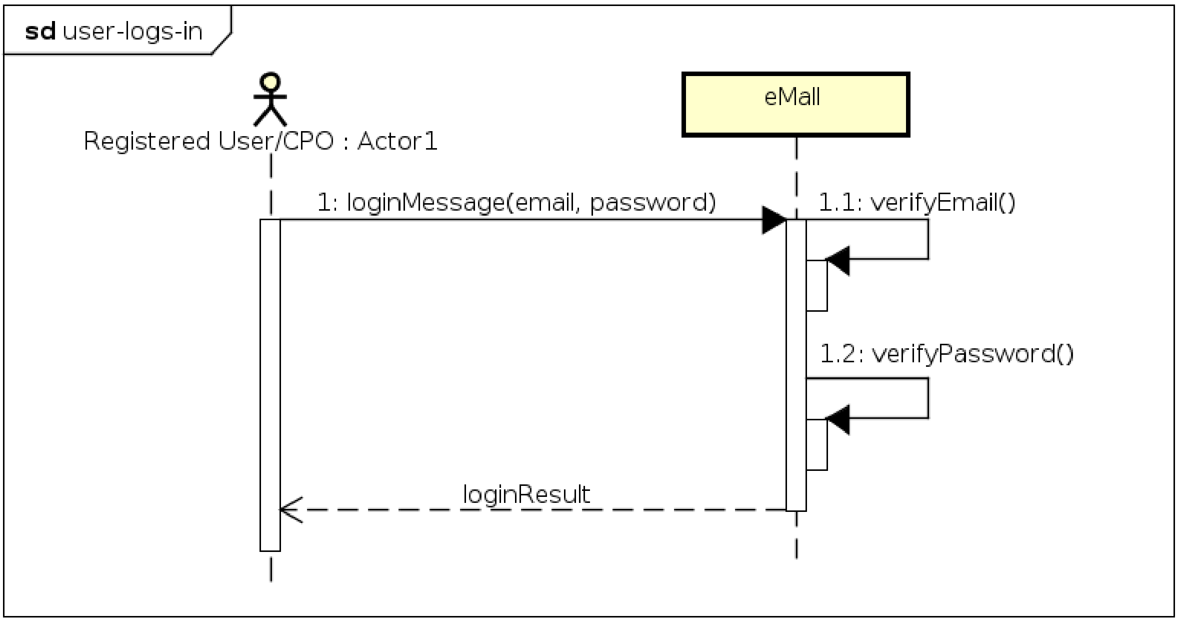


Figure 11: Login into eMall sequence

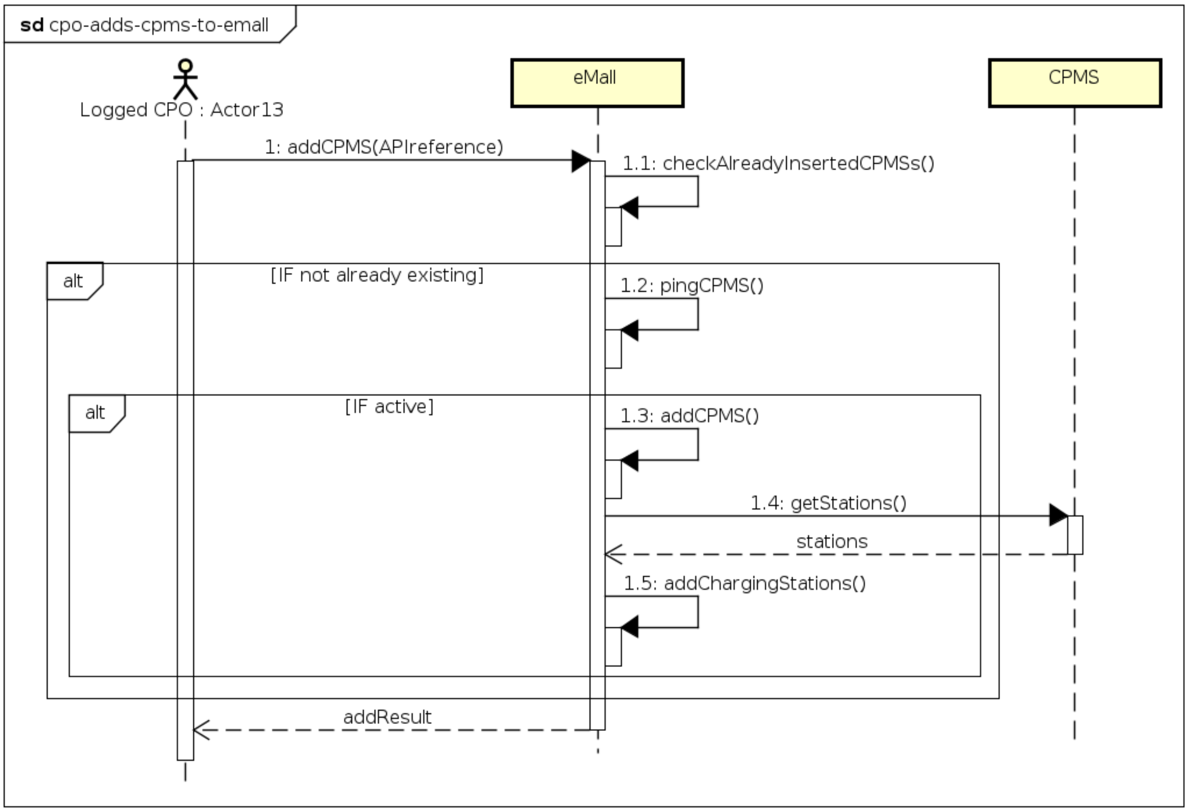


Figure 12: CPO adds a CPMS into eMall

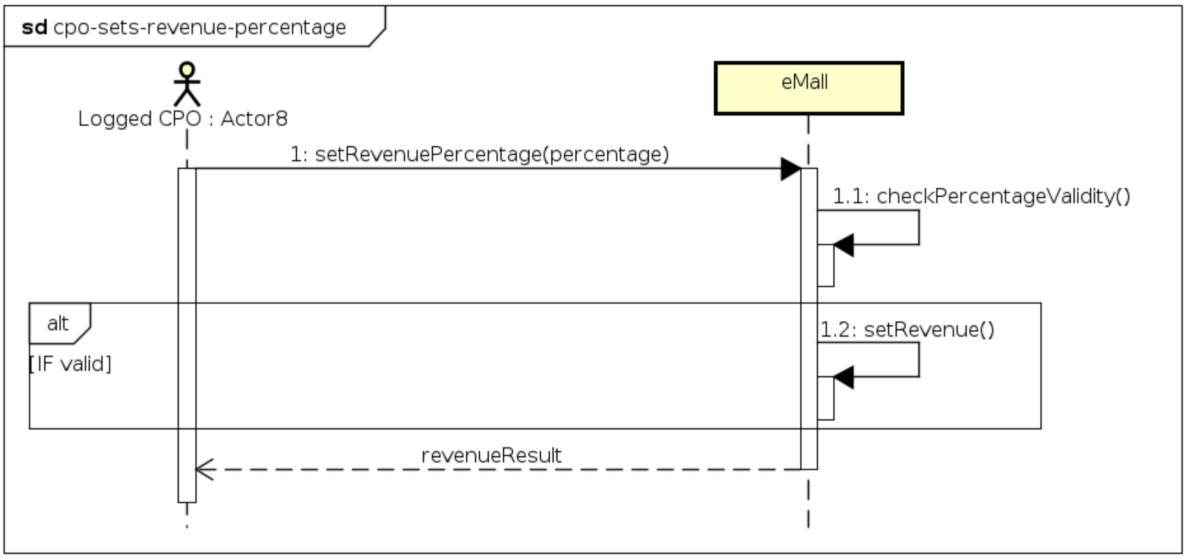


Figure 13: CPO sets the revenue percentage

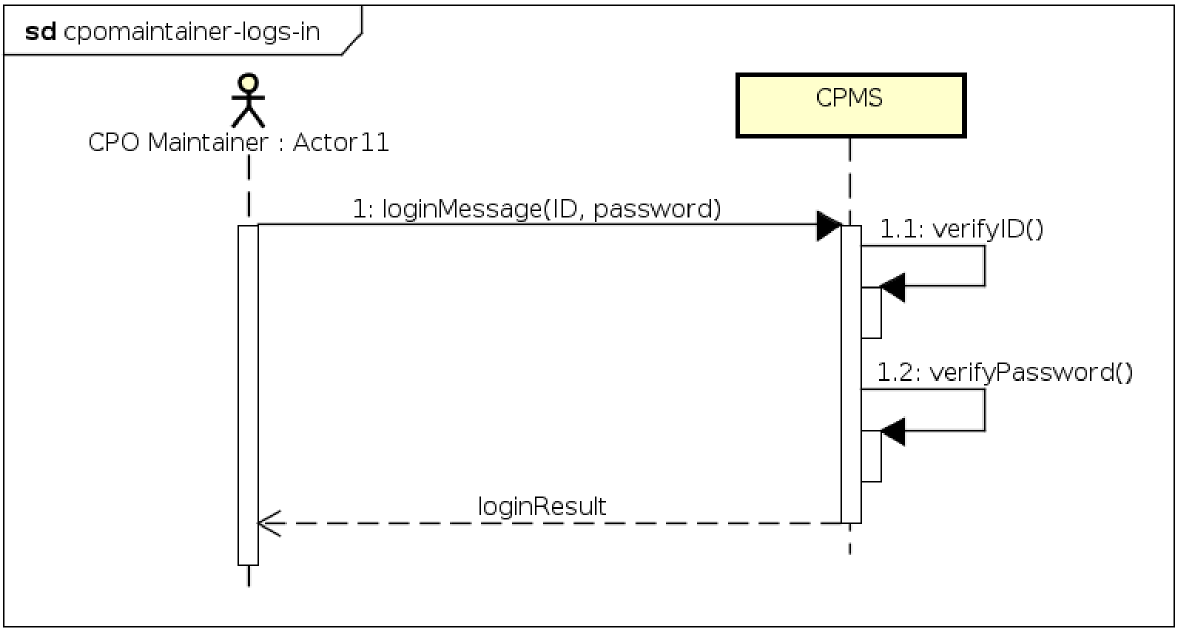


Figure 14: CPO maintainer logs into CPMS

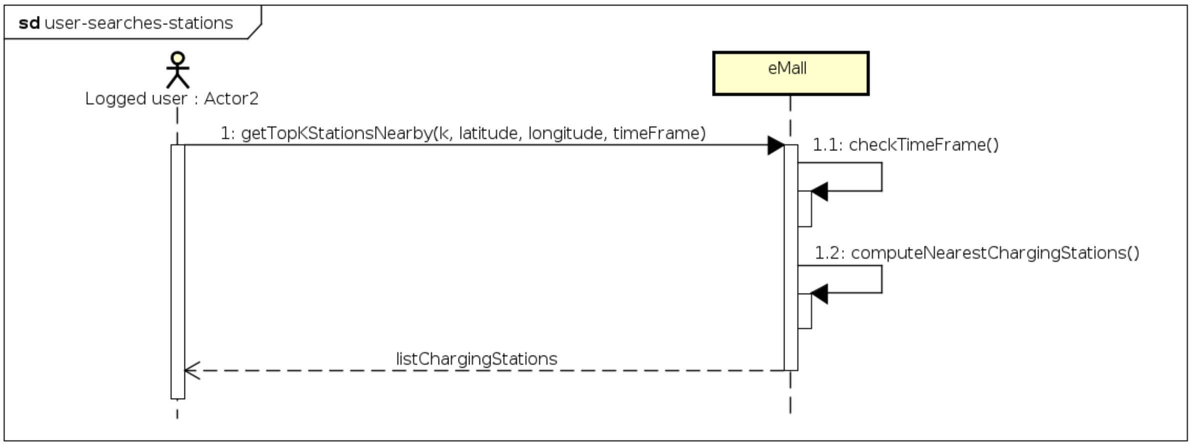


Figure 15: Get the nearby charging stations

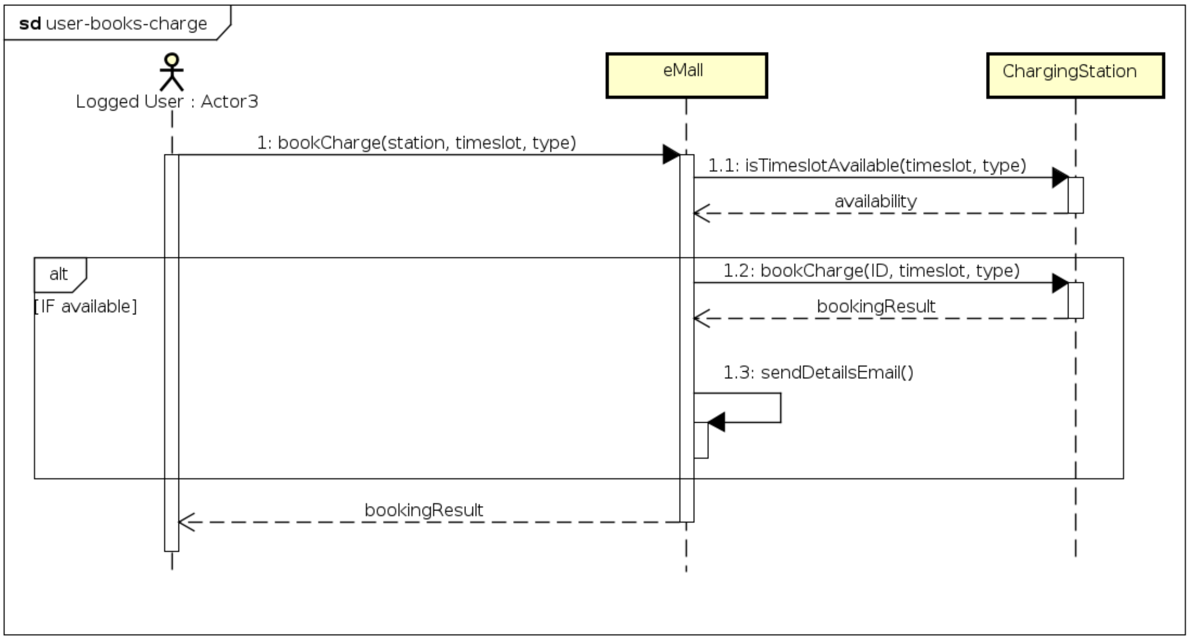


Figure 16: Book a charge sequence

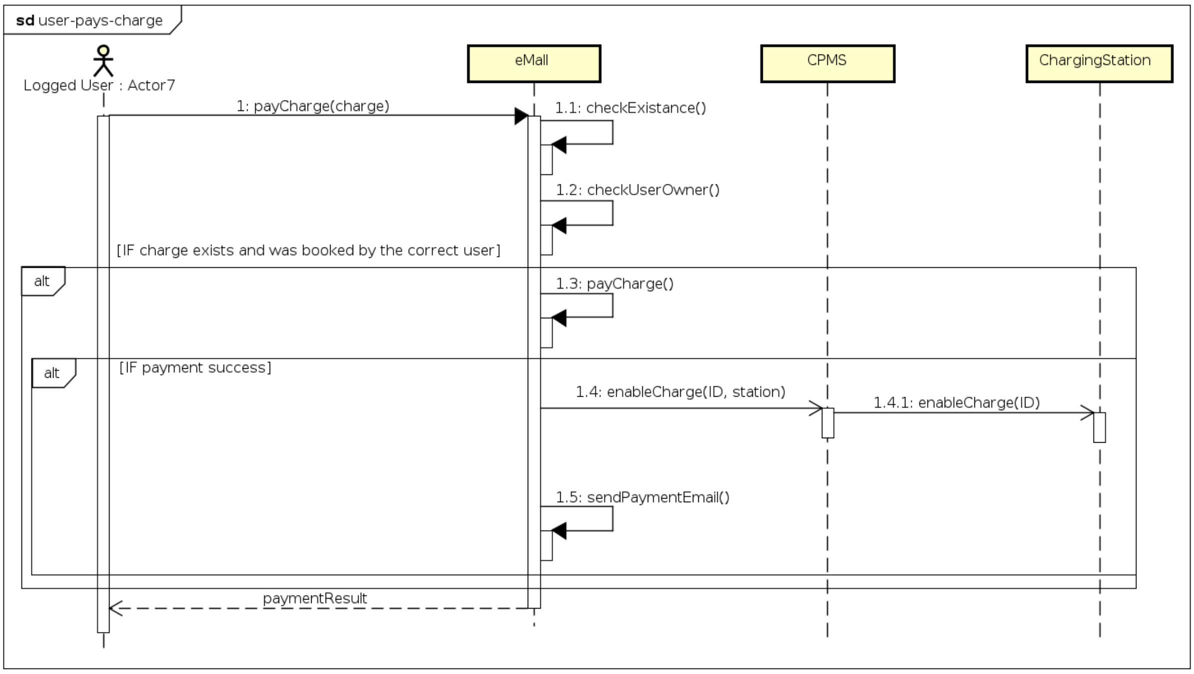


Figure 17: Pay a charge sequence

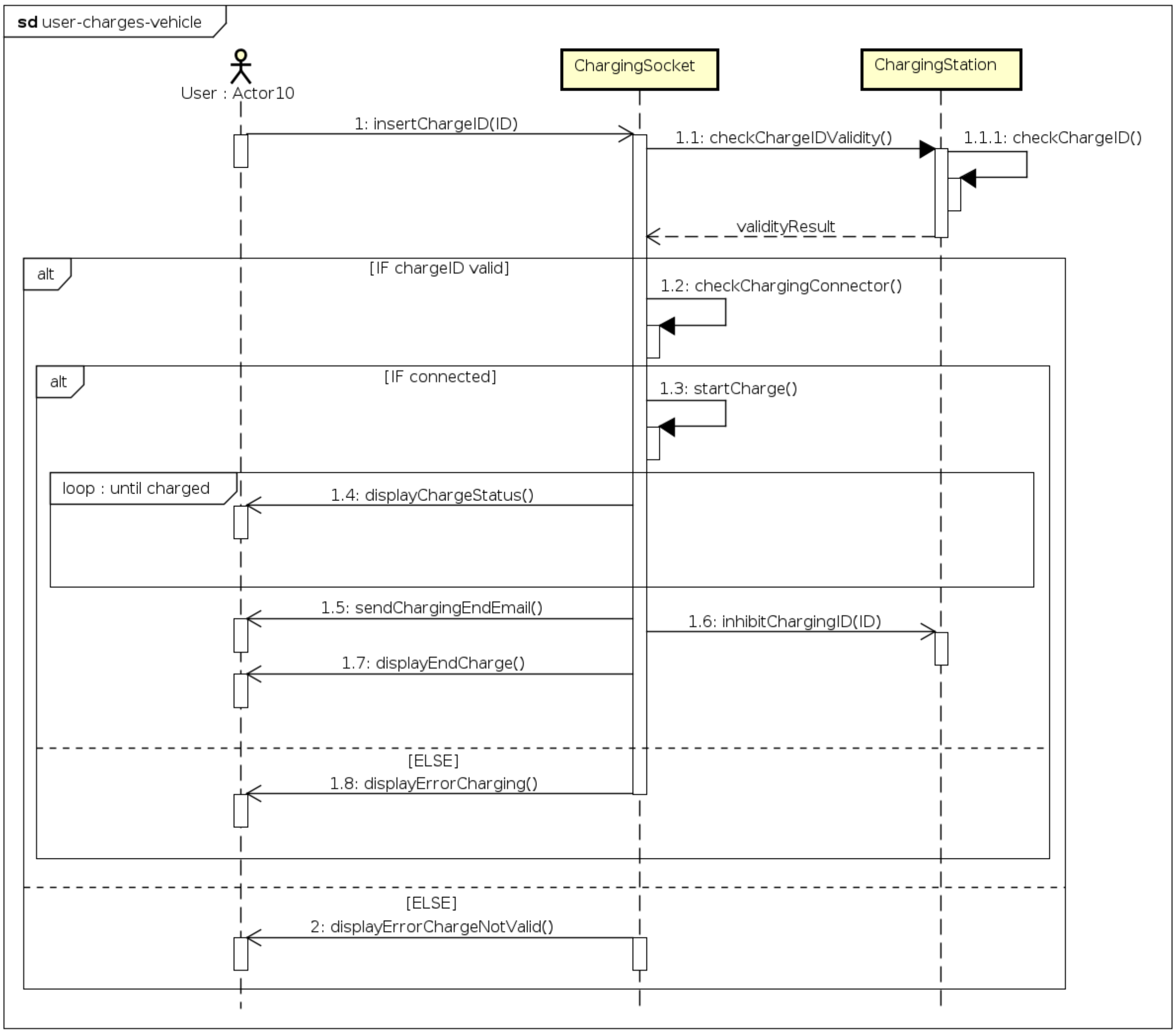


Figure 18: Perform a charge sequence

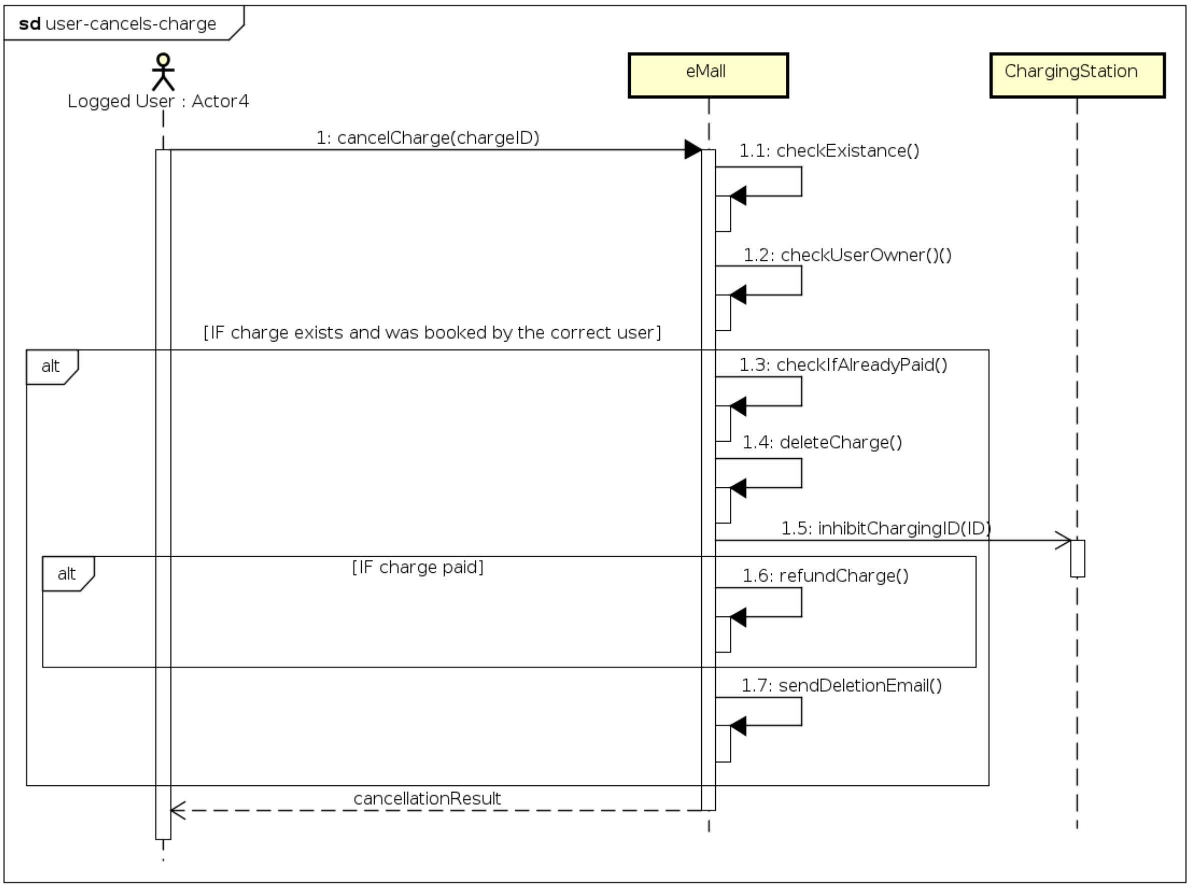


Figure 19: Cancel a charge sequence

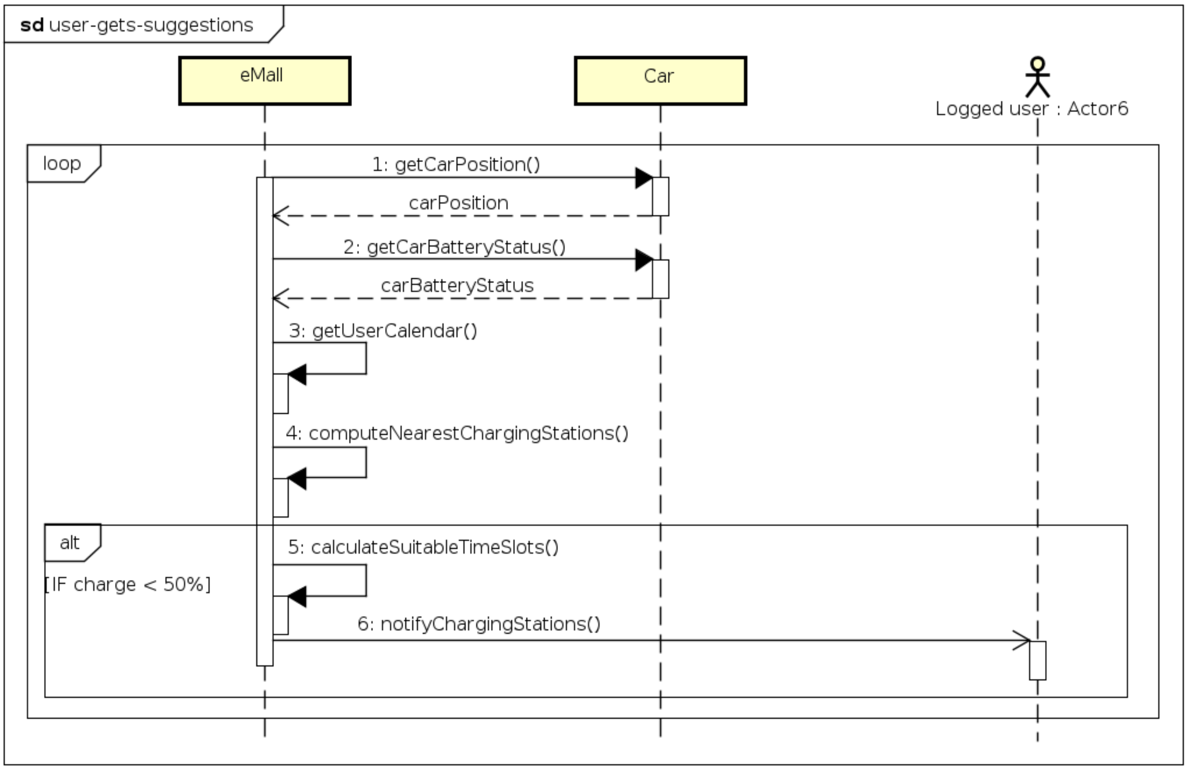


Figure 20: Charging suggestions via calendar sequence

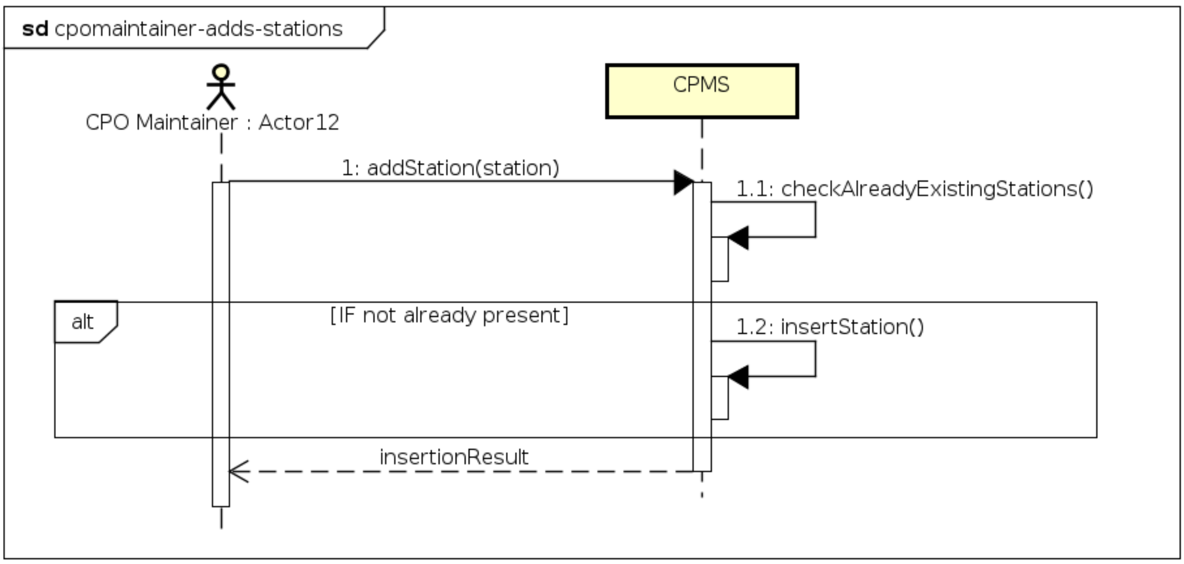


Figure 21: CPO maintainer adds stations to CPMS

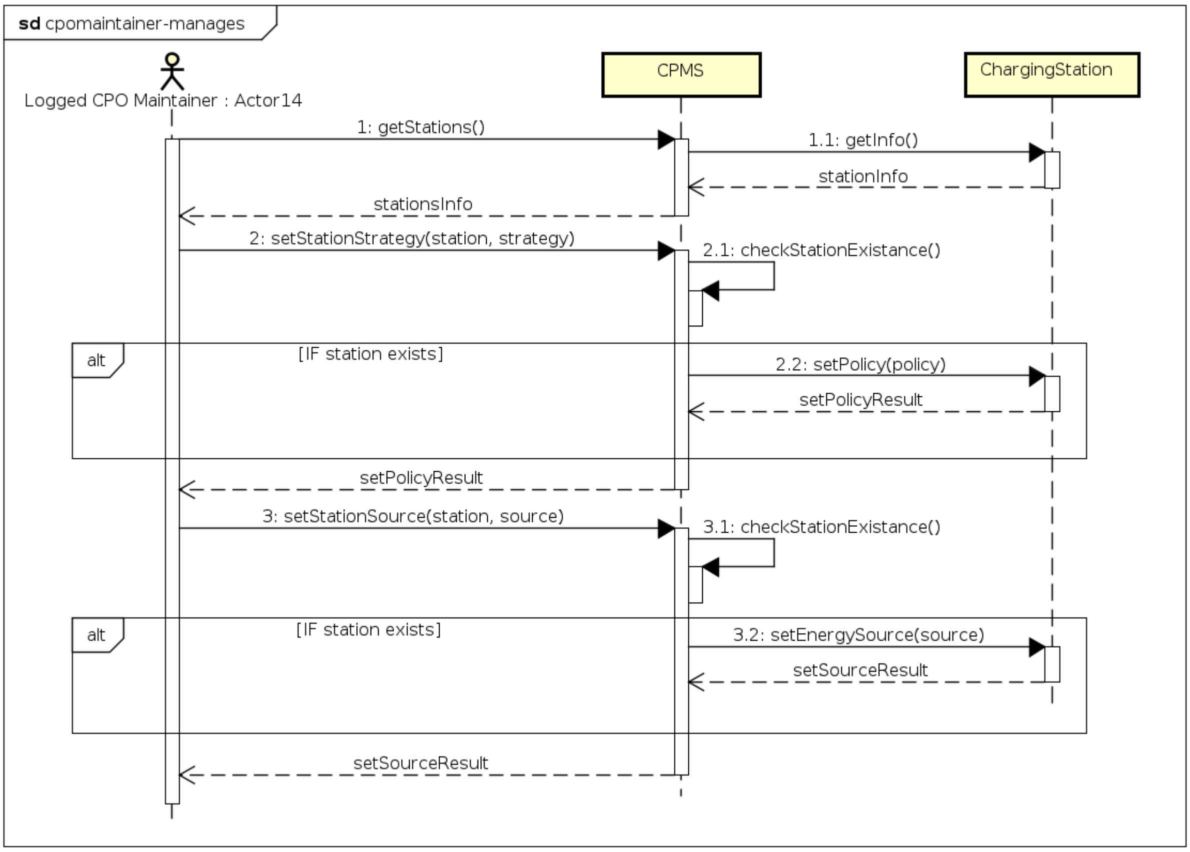


Figure 22: CPO maintainer manages a CPMS

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 much as possible, trying to make the client responsible of the heaviest loads.

These requirements are balanced considering the actual percentage of electric vehicle users in Italy [0.56%] (corresponds to about 330k users) and a level of confidence of 3.

3.4. Design constraints

From this point we consider only the user side constraints as they represent the largest share of the system use base.

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 user side 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.

3.5. Software system attributes

3.5.1. Reliability

The system shall be fail safe, while the actual service can behave slower than expected it shall be still consistent with the results. To do so the system shall be distributed data and performance wise, allowing a scalability factor while being open for maintenance without completely experiencing downtime. 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 a period of downtime would be detrimental, eMall has to prefer the availability over the 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 personal data, it has to abide the GDPR law; thus an encryption of the user's password must be adopted and the access to the user's data must be restricted only to himself. It is important that nobody else, not even the system administrator, can access the user's data in compliance of the privacy laws.

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. Thus it would not be complicated to maintain a single or a restricted amount of nodes per time. This way the users would only experience at worst slowdowns but never downtime.

3.5.5. Portability

The system should be as cross platform as possible to increase the maintainability over different type of platforms.

4. Formal Analysis Using Alloy

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

- Static Analysis;
- Dynamic Analysis;
- Assertions;
- Word Generation;

For this analysis the following assumptions have been considered:

- The Float type (not defined) represents 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 properties are considered, so the attributes of basic types (such as Int,float,boolean,Data etc...) are not considered. This decision has been taken to simplify the model view and coding. Most platforms that could be used to implement the system, already support this or similar types of data.
As a guideline the types are written only in the declarations inside a comment; they are defined by unimplemented interfaces and their ranges are specified; this types are not considered in the rest of the document.

```
module eMall

//only CPMS used in the system are added

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




sig CPO{
    cpms: set CPMS
    //name: one String
    //email: one Email
    //pIVA: one Int
    //password: one String
}


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

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

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

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

sig ChargingStation{
    position:one Location,
    //batteryKWh: one Int,
    sockets: set ChargingSocket,
    strategy: one Strategy
    //bookedCharges: one Map
}
//{  batteryPresent.isTrue implies inRange[batteryKWh, 0, 1000]}

sig ChargingSocket{
    chargingType: one ChargingType,
    //available: one Bool,
    //maximumPowerAmount: one Int,
    energySource:one EnergySource
}
//{ inRange[maximumPowerAmount, 0, 1000]}

sig Charge{
    //paid: one Bool,
```

```
    station: one ChargingStation,
    user: one DefaultUser
    //confirmationId; one String,
    //amount: one Int,
    //date: one Date
}

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{
    //costPerKw: one Float
}
//{ inRange[costPerKw, 0, 10000]}
sig Battery extends EnergySource{
//capacity: one Int
}
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 uniqueMailForCPO{
    //no disjoint c1,c2: CPO | c1.mail = c2.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 uniqueCPMSForStation{
    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 DefaultUser.vehicles}

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

fact noUserWithoutEMSP{
    all u:DefaultUser|  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 allChargeAreFromChargingStationInTheEMSP{
    all e:EMSP,s:e.charges.station | s in e.cpos.cpms.stations }

fact maintainersMaintainStationOfTheSameCPO{
    all m:Maintainer, c1,c2:CPO|
    (not c1=c2 and m in c1.cpms.maintainers) implies
    m not in c2.cpms.maintainers }

fact chargingStationThatChargeHasToHaveAtLeastOneSocket{
    all c:ChargingStation | c in Charge.station implies #c.sockets>0}
```

4.2. Dynamic Programming

In this part the major operations are described and run; as a convention old represents the version before the execution of the predicate, while the new is the version after the execution.
The picture here shown are cut to emphasize the predicate result.

4.2.1. User books a charge

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

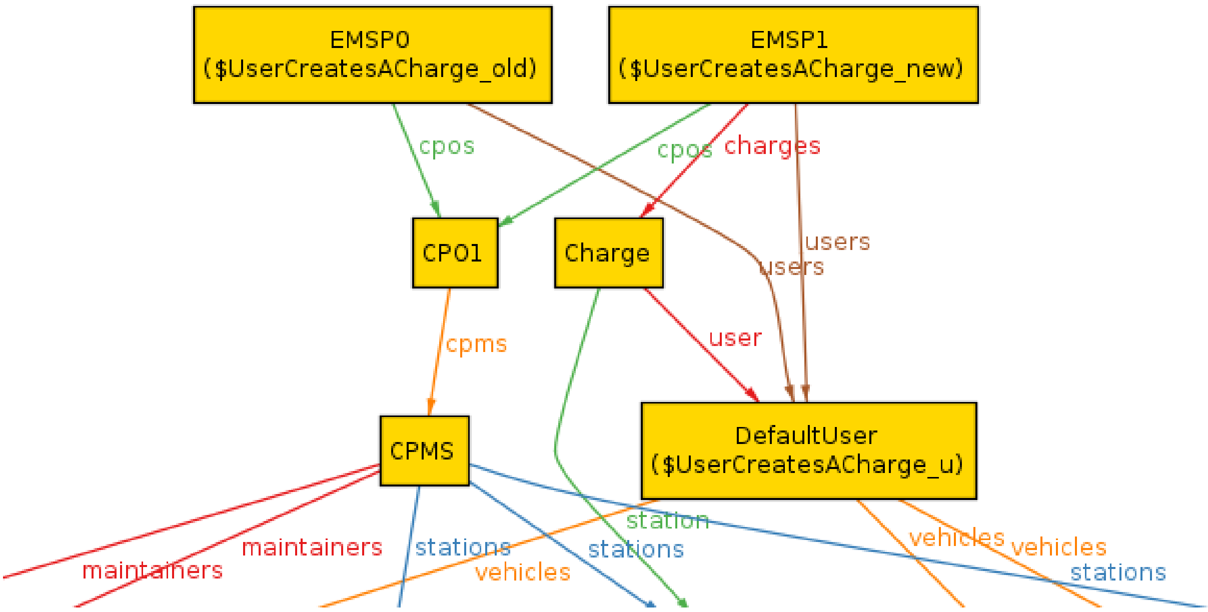


Figure 23: Added Charge

4.2.2. CPO subscribe to EMSP

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

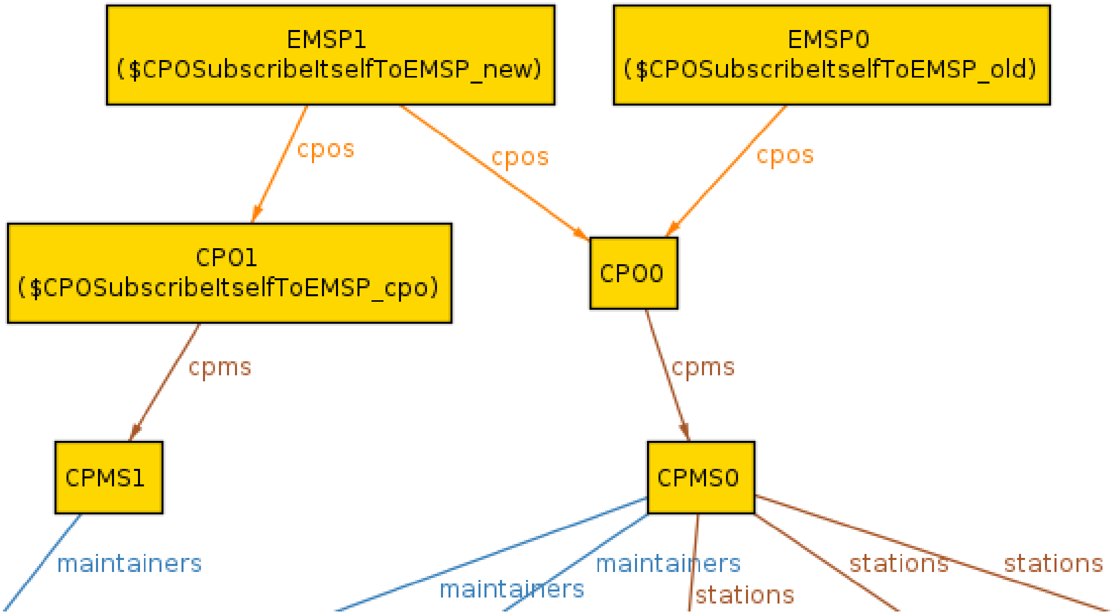


Figure 24: CPO subscribed

4.2.3. CPO add CPMS

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

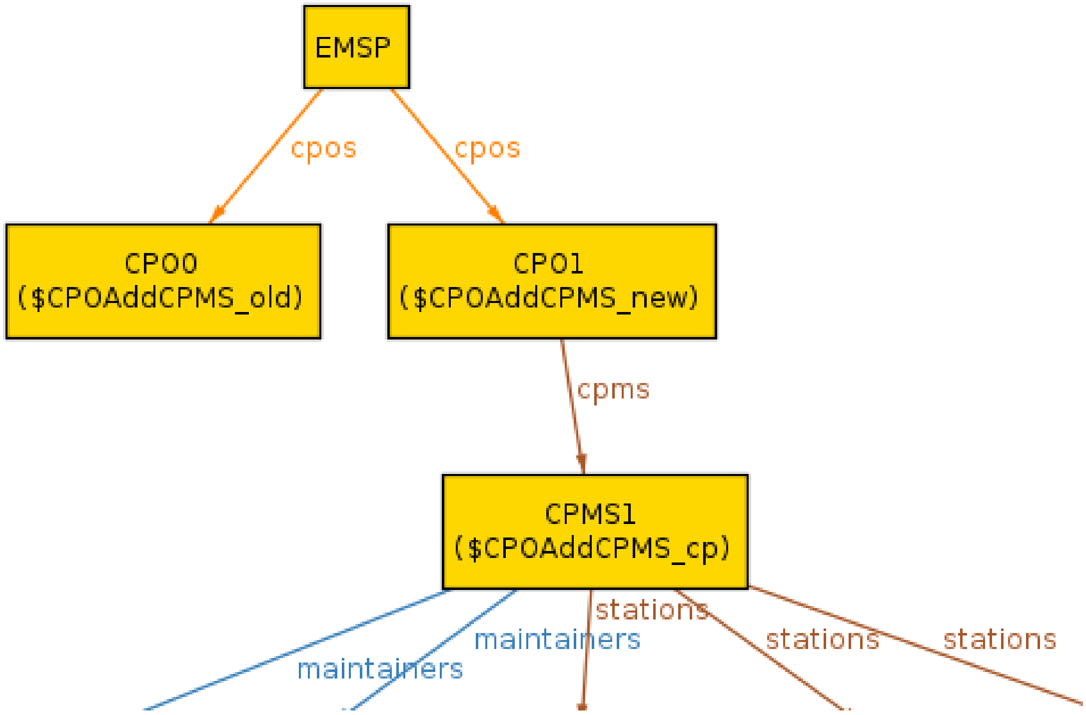


Figure 25: Added CPMS

4.2.4. CPO remove CPMS

Only one remove is shown since they are logically identical to their corresponding adds.

```
pred CPORemoveCPMS(new,old:CPO,cp:CPMS){
    not (old = new)
    cp in old.cpms
    new.cpms=old.cpms-cp
}
run CPORemoveCPMS for 3 but exactly 2 CPO, exactly 2 CPMS
```

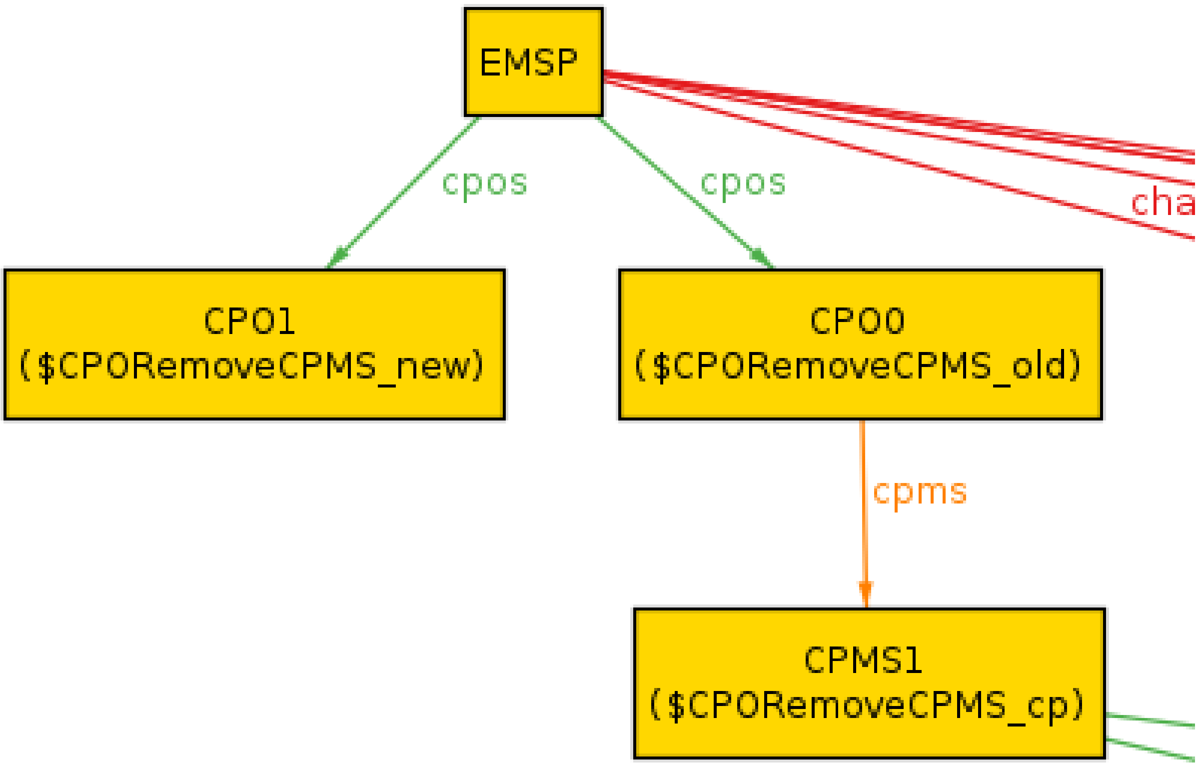


Figure 26: Removed CPMS

4.2.5. CPO add mantainer to CPMS

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

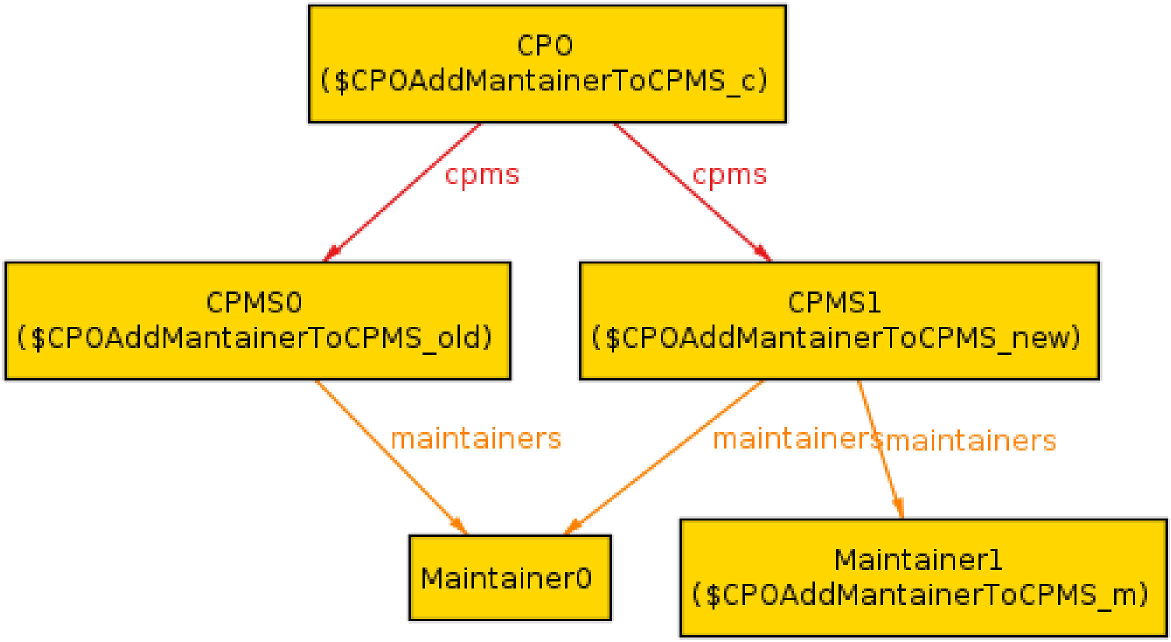


Figure 27: Added Maintainer

4.2.6. CPO add station to CPMS

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

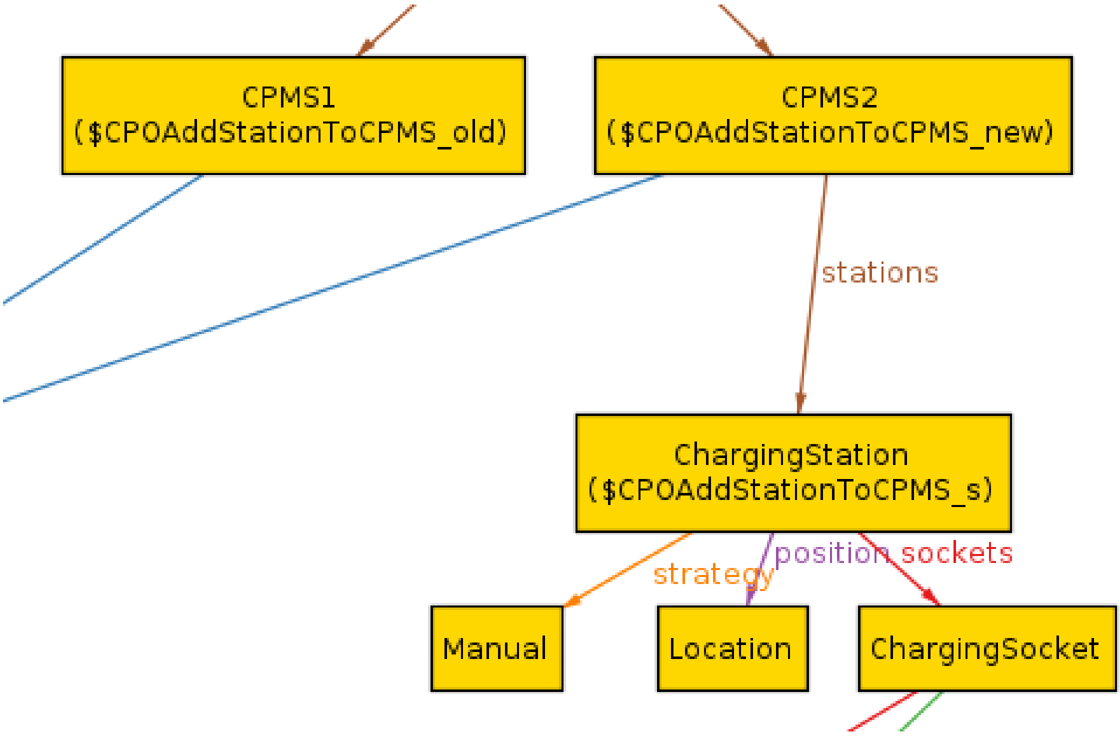


Figure 28: Added Station

4.2.7. CPO add socket to station

The following code does not solve; the add pattern used here consist in creating a new instance representing the status after the execution of the predicate. As a fact a socket can exist only in one station rendering this function inconsistent; this is due a limitation of the pattern, not the model, so in this document it is considered valid.

```
pred CPOAddSocketToStation
(c:CPO,cp:CPMS, new,old:ChargingStation,sk:ChargingSocket)
{
    not (new = old)
    cp in c.cpms
    new in cp.stations
    old in cp.stations
    new.position=old.position
    new.strategy=old.strategy
    new.sockets=old.sockets+sk
}
```

4.3. Assertions

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

```
//Asserts

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 uniqueCPMSForStationCheck{
    no disjoint c1,c2: CPMS, s:ChargingStation |
    s in c1.stations and s in c2.stations}
check uniqueCPMSForStationCheck 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.sockets)}
check socketOnlyOneStationCheck for 10

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

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

assert noUserWithoutEMSP{
    all u:DefaultUser|  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 maintainersMaintainStationOfTheSameCPOCheck{
    all m:Maintainer, c1,c2:CPO|
    (not c1=c2 and m in c1.cpms.maintainers)
    implies m not in c2.cpms.maintainers }
```

```
check maintainersMaintainStationOfTheSameCPOCheck for 10

assert chargingStationThatChargeHasToHaveAtLeastOneSocketCheck{
    all c:ChargingStation | c in Charge.station implies #c.sockets>0}
check chargingStationThatChargeHasToHaveAtLeastOneSocketCheck 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. uniqueCPMSForStationCheck 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. maintainersMantainStationOfTheSameCPOCheck may be valid.
#17: No counterexample found. chargingStationThatChargeHasToHaveAtLeastOneSocketCheck may be valid.
```

Figure 29: Assertion output

4.4. Word Generation

Here is the code of the word generation.

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

And the generated word.

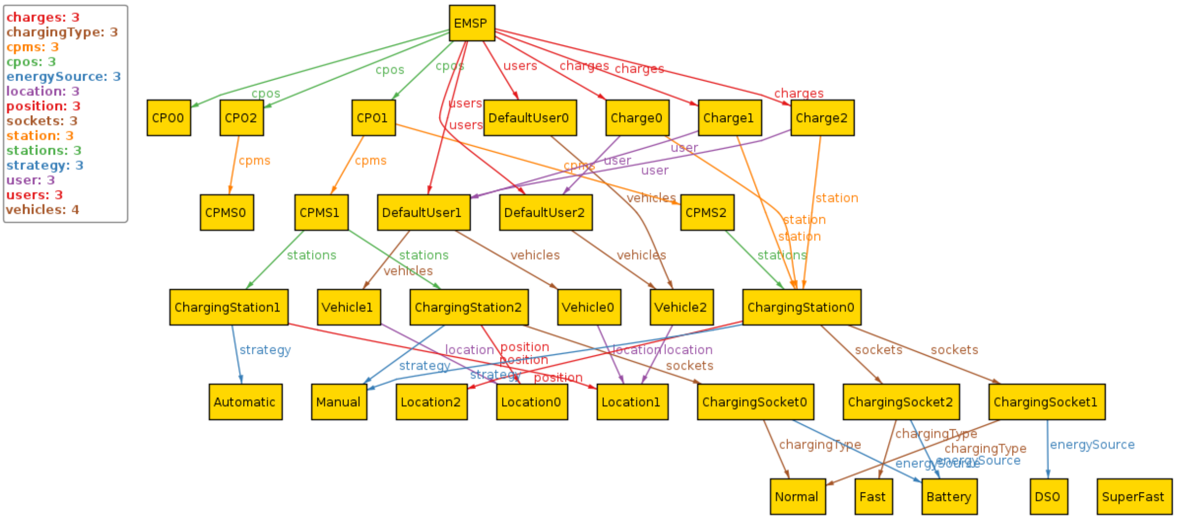


Figure 30: Generated Word

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
- 22/12/2022: 14:00 - 20:00 Federico, Matteo
- 22/12/2022: 16:30 - 17:30 Emilio
- 22/12/2022: 22:30 - 01:30 Emilio
- 23/12/2022: 09:30 - 13:30 Emilio
- 23/12/2022: 13:30 - 14:30 Matteo
- 23/12/2022: 13:00 - 16:00 Federico
- 23/12/2022: 15:30 - 20:00 Emilio
- 23/12/2022: 20:30 - 23:00 Federico, Emilio

6. References

- **RACS and RAPS**: https://arxiv.org/pdf/cs/9912010.pdf
- **Partita IVA**: https://www.agenziaentrate.gov.it/portale/web/guest/agenzia/amministrazione-trasparente/servizi-erogati/carta-servizi/i-nostri-servizi/area-identificazione-del-contribuente/partita-iva
- **Electric transport percentage**: https://ourworldindata.org/transport