AY 2022/2023

POLITECNICO
MILANO 1863

RASD: Requirement Analysis and Specification Document

Marcello De Salvo Riccardo Grossoni
Francesco Dubini

Professor
Elisabetta Di Nitto

Version 0.1
December 11, 2022

Contents

1 Introduction

Electric mobility (e-Mobility) is a way to reduce the carbon footprint caused by motorized vehicles in urban and sub-urban areas.
Comfortably knowing how to fit the charging process into one's daily schedule is a fundamental step towards that goal.

1.1 Purpose

In the last 3 years the number of electric vehicles has doubled and, with the proposed european legislation on banning diesel fuel by 2035, the future seems to be full-electric. This rapid change requires a better and more efficient infrastructure to serve this ever increasing demand.

eMall (e-Mobility for all) aims to ease the charging process for the users through the *e-Mobility Service Provider's* (eMSP) platform, providing all the needed end-users' services and by actively communicating with multiple *Charging Point Operators'* (CPOs) *Management Systems* (CPMS). The eMall platform will also provide a dashboard for the *Charging Point Operators' workers*(CPOW) to change some settings that can also be done automatically by the CPMS.

1.1.1 Goals

User goals

- **(G1) Know about the charging stations nearby, including their cost and any special offer they have.**
- **(G2) Book a charge in a specific charging station for a certain timeframe.**
  description
- **(G3) Start the charging process at a certain station.**
- **(G4) Pay for the obtained service.**

eMSP goals

- **(G5) Be able to communicate with multiple CPMS.**
- **(G6) Suggest the user convenient charging timeframes based on his upcoming schedule.**
- **(G7) Notify the user when the charging process is finished.**

CPMS goals

- **(G8) Be able to communicate with multiple eMSPs.**
- **(G9) Be able to communicate with at least one DSO.**
- **(G10) Communicate the location and "external" status of a charging station.**
- **(G11) Know the "internal" status of a charging station .**
- **(G12) Start charging a vehicle according to the amount of power supplied by the socket, and monitor the charging process to infer when the battery is full.**
- **(G13) Acquire from the DSOs information about the current price of energy.**
- **(G14) Decide from which DSO to acquire energy (if more than one is available).**
- **(G15) Dynamically decide where to get energy for charging (station battery, DSO, or a mix according to availability and cost).**

CPOW goals

- **(G16) Change the source of energy between batteries and grid.**
- **(G17) Insert special and limited offers.**
- **(G18) Change the price of a charging station.**

1.2 Scope

Shared Phenomena

ID	Phenomenom	Controller
S1	User registers through the application	World
S2	User logs into the application	World
S3	User gets notified by the system about the status of the charge	Machine
S4	User is suggested by the system to go and charge the vehicle, depending on the status of the battery, his daily schedule and the charging type	Machine
S5	User is presented with special offers made available by some CPOs	Machine
S6	The system shows the availability of nearby charging stations	Machine
S7	User pays the cost of the charge through the application	World
S8	CPOW inserts a new discount into the system	World

World Phenomena

ID	Phenomenom
W1	Power outage in a station nearby area
W2	Physical problem in the charging station
W3	Physical problem in the electric vehicle
W4	Unexpected changes in the user daily schedule
W5	Person other than the user is driving the car

1.3 Definitions, acronyms, abbreviations

Definitions

- **User**: any electric car owner.
- **e-Mobility Service Providers**: company offering an electric vehicle charging service to drivers by providing access to multiple charging points around a geographic area.
- **Charging Point Operator**: charging point stations owner.
- **Charge Point Management System**: charging Point Operator's IT infrastructure. Handles the acquisition of energy from external Distribution System Operators and distributes it to the connected vehicles. It can also makes automatic decisions, such as the amount of energy to be used for each connected vehicle.
- **Distribution System Operator**: entity responsible for distributing and managing energy from the generation sources to the final consumers.
- **Open Charge Point Interface**: open protocol used for connections between charge station operators and service providers.
- **Open Charge Point Protocol**: open protocol used for connections between charge stations and their charge point management system.
- **DIN SPEC 70121**: Standard for digital communication between a charging station and an electric vehicle.
- **ISO 15118**: Alternative standard for digital communication between a charging station and an electric vehicle.

Acronyms

- **EV**: Electric vehicle
- **eMSP**: e-Mobility Service Providers
- **CPO**: Charging Point Operator

- **CPMS**: Charge Point Management System
- **OCPI**: Open Charge Point Interface
- **OCPP**: Open Charge Point Protocol
- **DSO**: (3rd party) Distribution System Operator
- **API**: Application Programming Interface
- **UML**: Unified Modeling Language
- **CPOW**: Charging Point Operator Worker
- **BESS**: Battery Energy Storage System
- **RTC**: Real Time Communication

1.4 Revision history

- Version 0.1: Setup
  - Created first layout

1.5 Reference documents

- Specification document: "Assignment RDD AY 2022-2023"
- Alloy documentation: https://alloytools.org/documentation.html
- Data on number of electric vehicles: https://www.iea.org/data-and-statistics/charts/global-electric-car-stock-2010-2021

1.6 Document structure

- **Section 1**: introduces the problem, describes every goal of the project and gives an analysis of the world and shared phenomena.
- **Section 2**: gives an overall description of the project and all the interactions that will occur between the system and the final users, including a list of possible scenarios and a description of all the actors involved. It provides also an UML class diagram that will be used as a reference point for the developers.
- **Section 3**: includes all the project's requirements and an in-depth description everything presented in Section 2.
- **Section 4**: shows the Alloy model defined for this project.

2 Overall Description

2.1 Product perspective

In this section we mainly describe some typical scenarios, all the *Product Functions (2.2)* offered by the eMall system and the*UML(2.1.2)* class-diagram.

2.1.1 Scenarios

1. Mister Fontana has just bought an electric vehicle and wants to create an account for the application.

   - He opens the web app and clicks the "Sign Up" button
   - He inserts all the required information in the mandatory fields and presses the "Confirm" button
   - The system verifies that the mail has not been used before. After passing the verification a confirmation e-mail is sent to mister Fontana's mailbox
   - He checks his mailbox to see if he received the confimation e-mail

2. Mister Brambilla is about to finish a meeting and is about to have a 2 hour break. eMall detects that he's about to go on break, and it sends him an e-mail suggesting him to charge his car

   - He receives an e-mail suggesting him to charge his car
   - He logs on the eMall webapp
   - The eMall webapp suggests the closest chargers and the ones with eventual discounts
   - Brambilla selects his preferred charging station and books it
   - eMall prompts a link to help Brambilla get there
   - Brambilla opens the link and heads there

3. Mister Fontana is on vacation and since he's new to the area he wants to know which charging stations are nearby

   - He opens the web application and logs-In

- He navigates to the interactive map section
- He clicks the "current position" button
- He gets notified by the web application that his GPS position is not turned on
- He turns on the GPS and then searches all the nearby stations by simply clicking the "current position" button and viewing the map

4. Miss Sala will go to the hairdresser next sunday. She knows that in the parking lot near the hairdresser there's a charging station and she wants to make sure that a spot will be available that day, so she decides to book a spot for that charging station

    - She opens the web application and logs-In
    - She selects the "Book a spot" option
    - She compiles the requested fields with date, time of arrival and location and selects confirm
    - The system checks if a charging spot for that date and time is still available, in that case a confirmation message is shown and the booking is saved. If no spot is still available an "unsuccesful booking" message is shown and the booking is aborted
    - After successfully booking a spot the system sends a mail with a recap of the booking
    - Miss Sala after reaching the charging station on the booked time-frame unlocks the booked charging spot via the web app and can start the starting process

5. Mister Ferrari has booked a charging spot for saturday. He realizes that he has lost the keys to his car, and will not make it in time. While he searches for his keys he forgets to delete the reservation.

    - The system locks the charging spot while waiting for mister Ferrari
    - 5 minutes after the booked time, if the reservated user doesn't show up, the charging spot is unlocked and the reservation is cancelled

- Other users can now start using the previously reserved charging spot

6. Mister Rana wants to start the charging process and pay, since he's arrived at the booked station near his office

   - He opens the web application and logs-In
   - He navigates to the "booking list" section
   - He plugs-in the charging cable to the car
   - He selects the right station and clicks the "start charging" button
   - He's then asked to pay inside the web app and he pays with his credit card
   - He finally sees that the car has started to charge
   - When the charging ends he gets notified by the application

7. Mister Rossi wants to plan his trip to Rome with his electric car. He would also like to save money by selecting the cheapest charging stations along the itinerary.

   - He opens the web-application and he logs-In
   - He plans his trip inside the navigation system or the in-app map
   - He then sorts the charging stations by the charging price and the special offers
   - He books the cheaper ones that covers the right amount of distance from each other

8. Mister Lamborghini wants to delete his booked charge since his schedule changed and he can no longer make it there in time.

   - He opens the web-application and he logs-In
   - On the webapp he opens his "bookings list"
   - He then selects the one he wants to delete and clicks on "cancel booking"
   - An email of the succesful cancellation is sent and the system deletes the previous reservation

2.1.2 Class diagram

In the UML diagram there are described all the main entities in the system. We can see that the system is composed by 2 main actors: the *User* and the *CPO*.
The user owns an *E-car* and can book a charging spot, while the charging station is the entity that provides the charging service.
Data regarding the charging stations and the sockets are provided by the CPMS *OCPI API* which is used by the *Booking*, *Charging Process* and *Charging station* entities to retrive or send all the needed information, and that is implemented inside the CPMS subsystem.
There's also an entity for the *Socket* in order to store the information about the socket's status and his price, and another one for the *Offer* that can be applied to all the charging stations owned by the same CPO.
Information about the Users' daily schedule are represented inside the *Calendar Schedule* and the the *Event* entity, that use the *Calendar API* to retrive the user's appointments.
There's also an interface for the *Payment* that can be implemented by different payment methods.

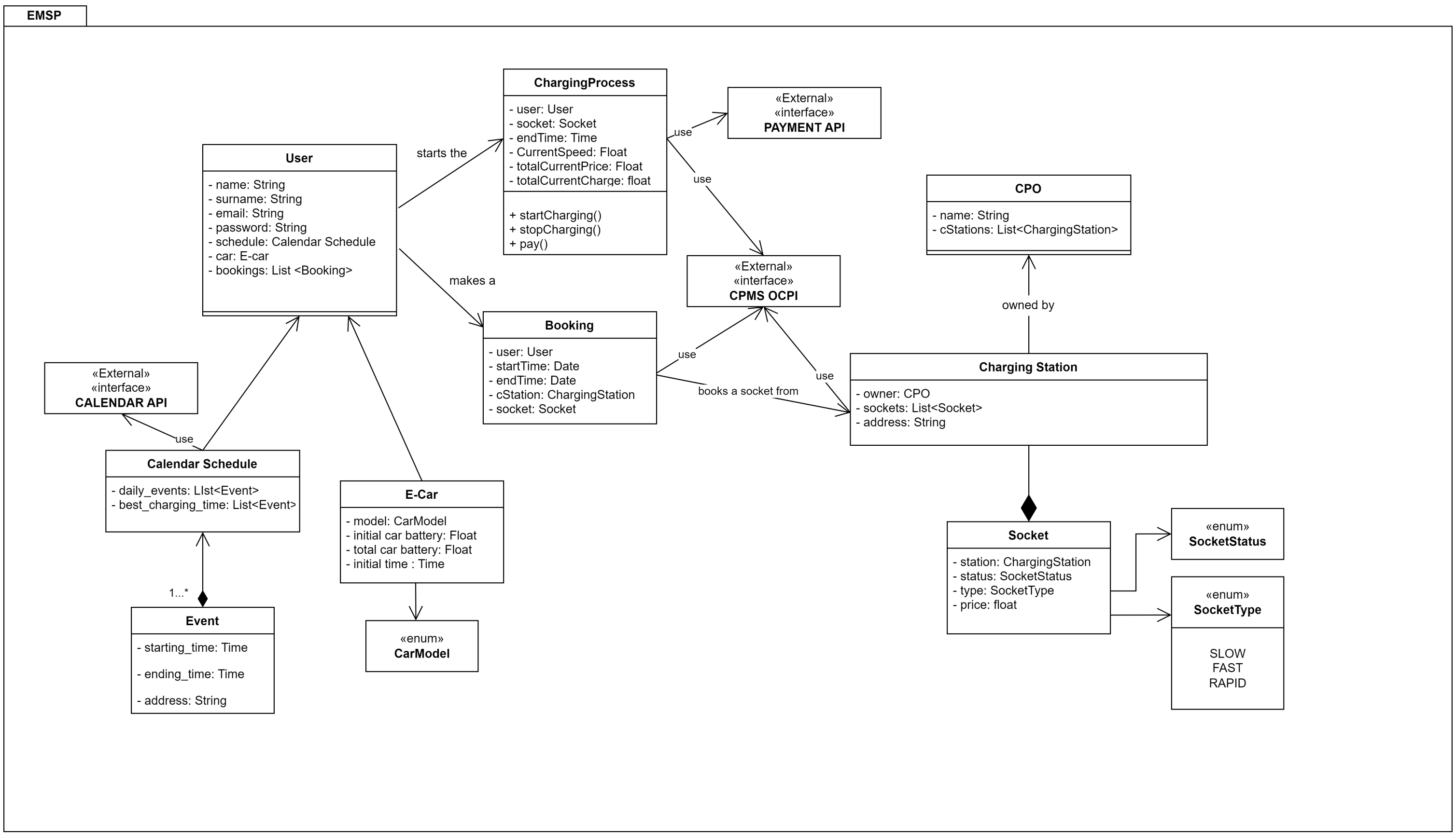

Figure 1: High-level UML - EMSP Subsystem

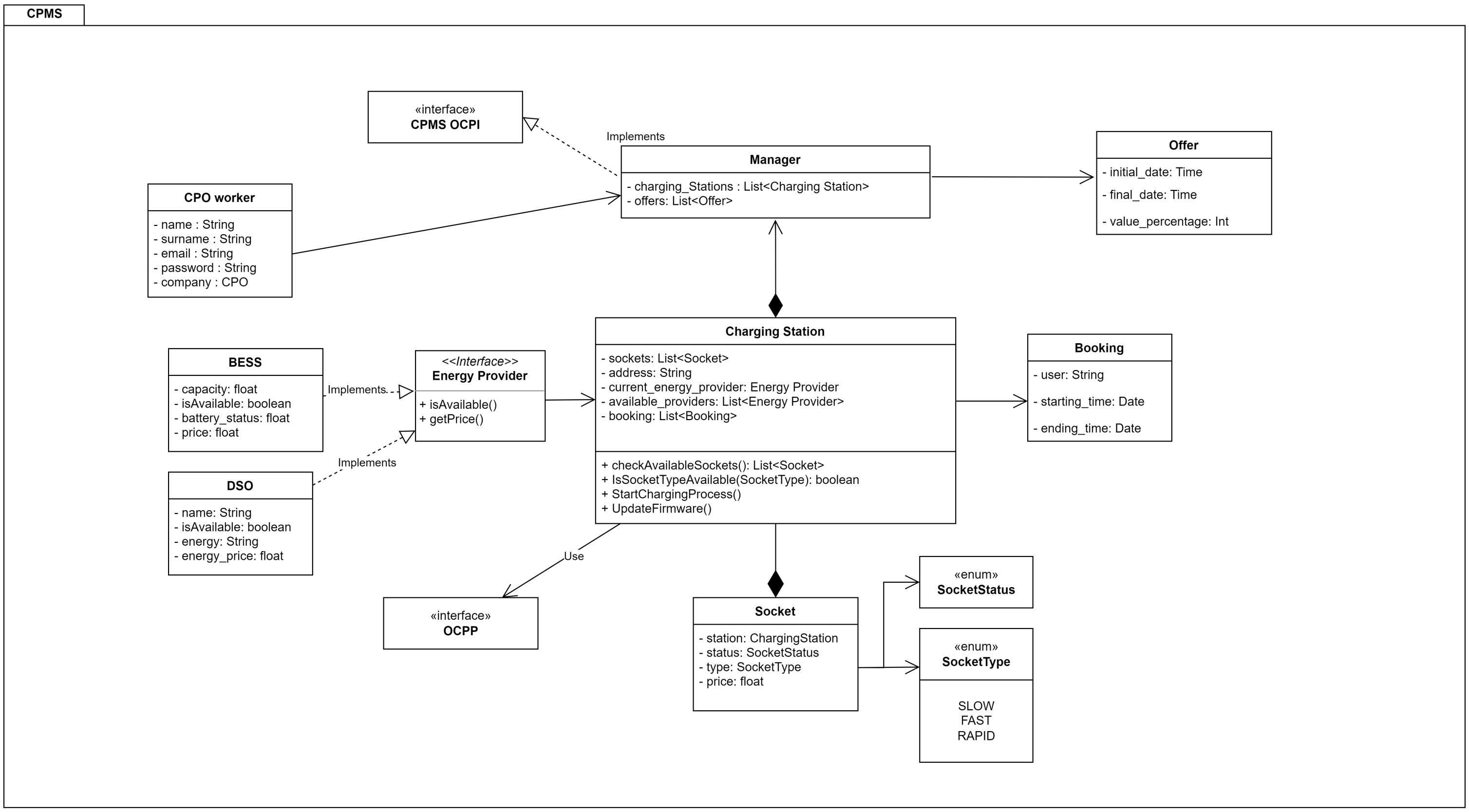

Figure 2: High-level UML - CPMS Subsystem

2.2 Product functions

2.2.1 Shared

- **Sign-up:** let the user sign-up thorugh an email and a password.

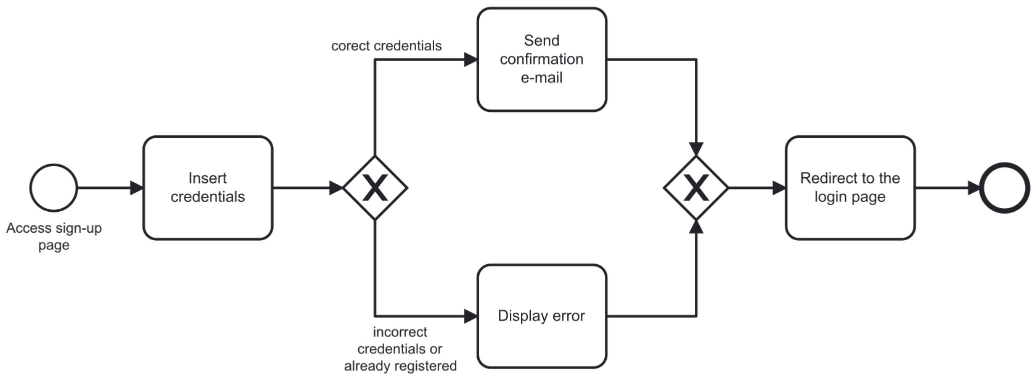

Figure 3: Sign Up BPMN

2.2.2 EMSP functions

- **Check nearby stations:** let the user check the nearby charging stations through an inter map and the GPS position.

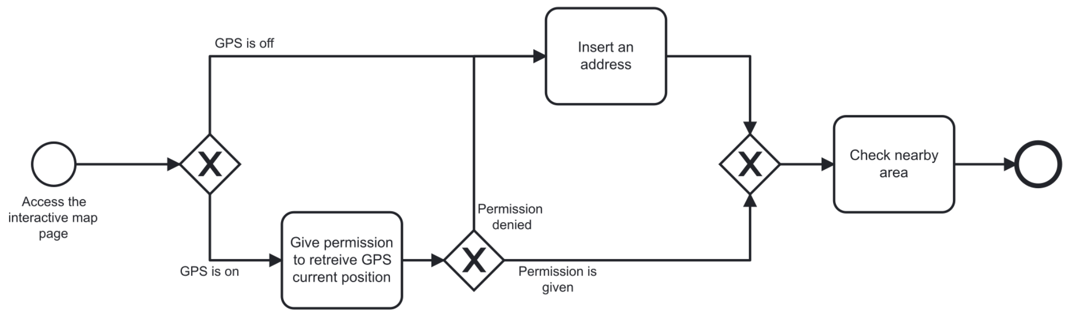

Figure 4: check nearby stations

- **User's schedule suggestions:** notify the user about the best charging stations to visit based on the user's schedule and their current price. The check would be done at least once a day based on the events planned for the next day.

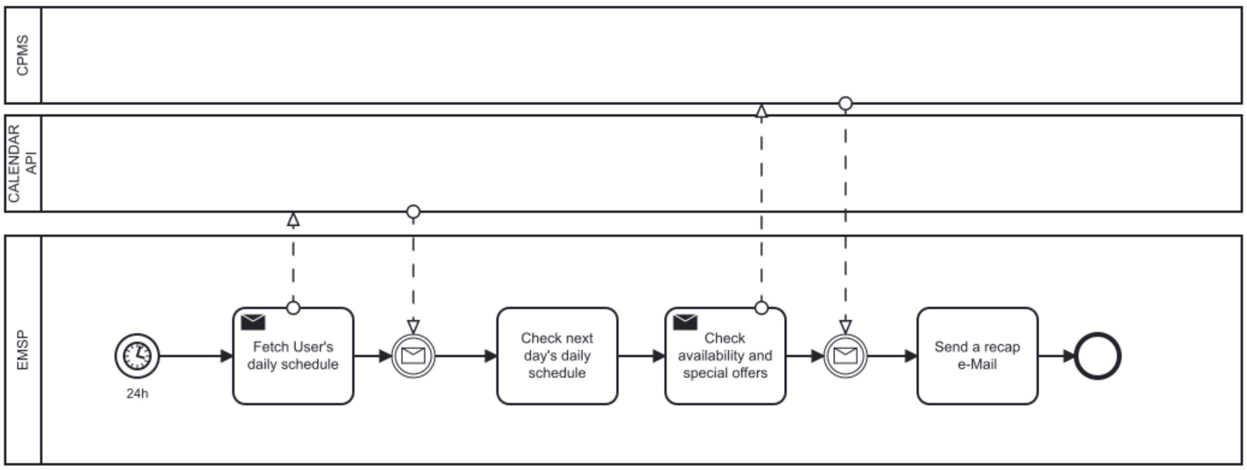

Figure 5: User's schedule suggestions

- **Battery and navigation system suggestions:** notify the user about the best charging stations to use based on the car's battery level and the current route.

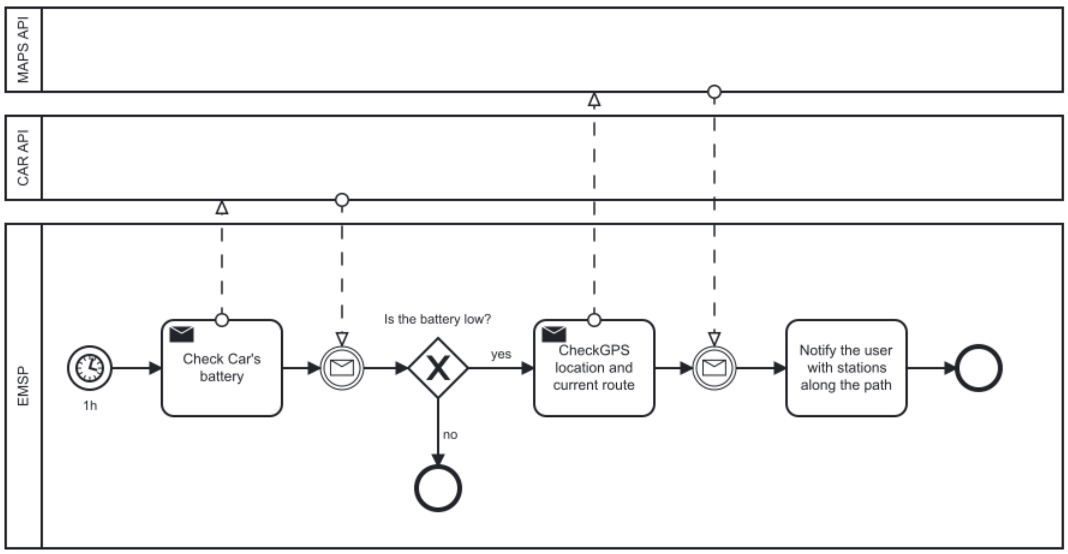

Figure 6: Battery and navigation suggestions

- **Booking a charging spot:** let the user book a charging spot for a specific timeframe. The user can also see the available charging spots and the price for each one.

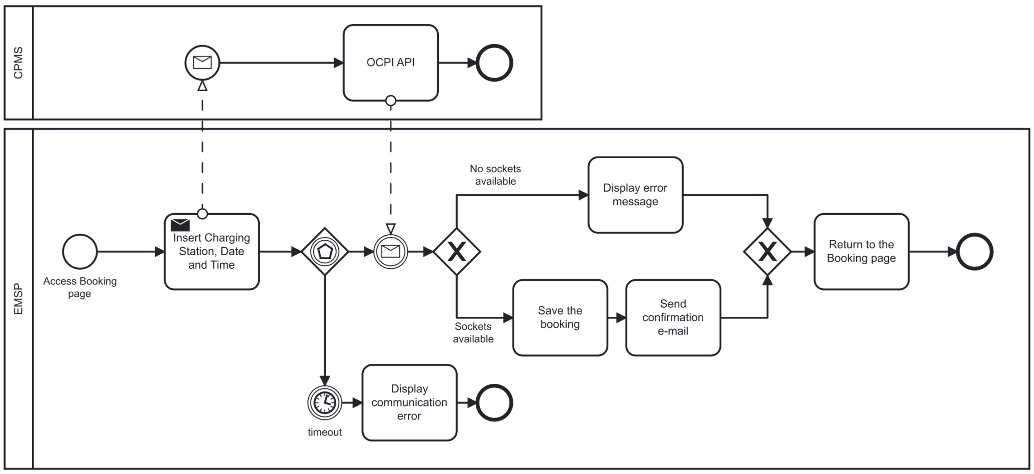

Figure 7: Booking a charging spot

- **Charging Process:** let the user start the charging process and pay for the service.

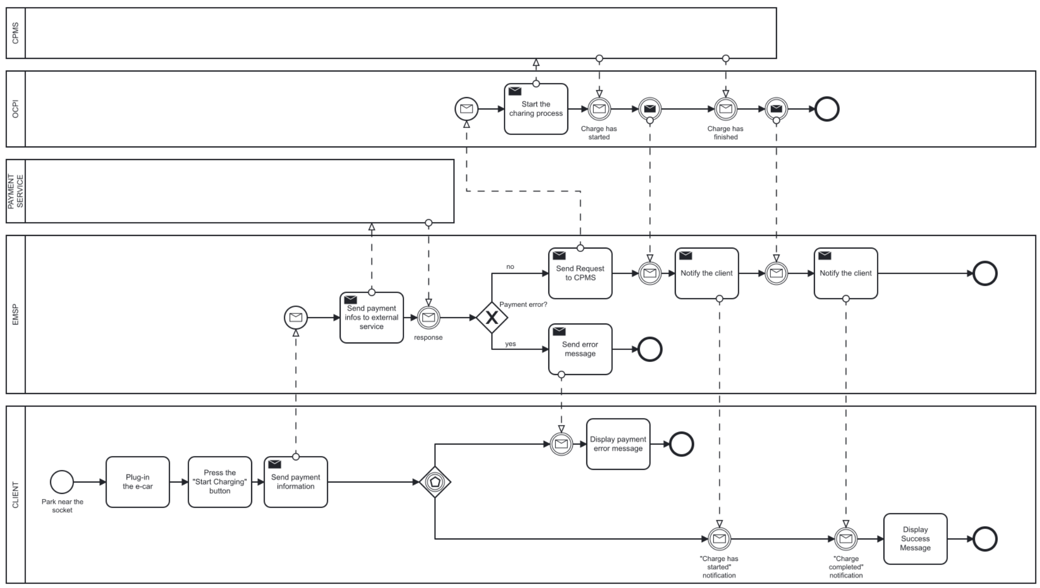

Figure 8: Charging Process

2.2.3 CPMS functions

- **Retrieve Station's status:** the CPMS queries the charging stations through the OCPP protocol and sends back the desired information to the emsp through the OCPI.

- **Retrieve DSO's status:** the CPMS queries the available DSOs through their communication interface about their current price.

- **Charging Process:** the CPMS manages the charging process by communicating with the socket and the EMSP.

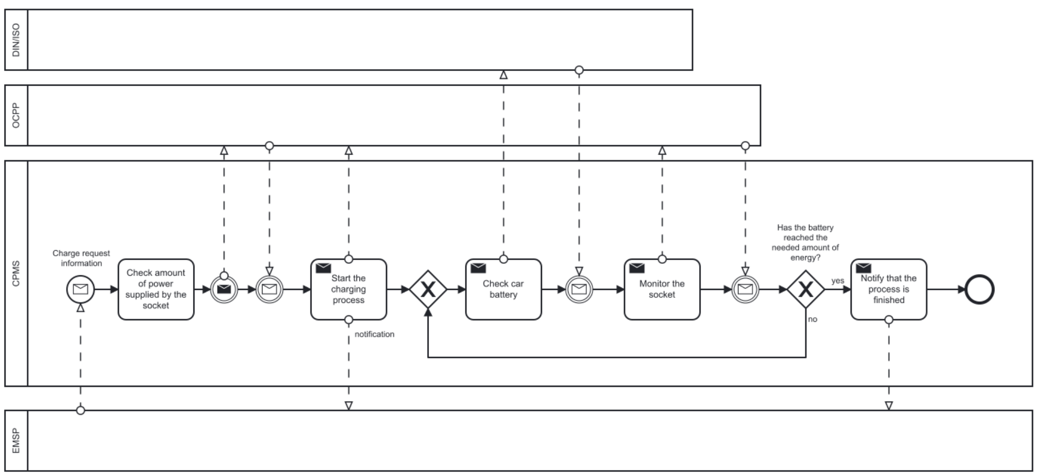

Figure 9: Charging Process

- **Dynamic selection of the source of energy:** the CPMS selects the best energy provider based on the price, the availability and the status of the internal battery. It sorts the energy providers by price, then if the internal battery is full, it selects the cheapest energy provider. If the internal battery is not full, it selects a mix of the cheapest energy provider and the internal battery based on the battery level.

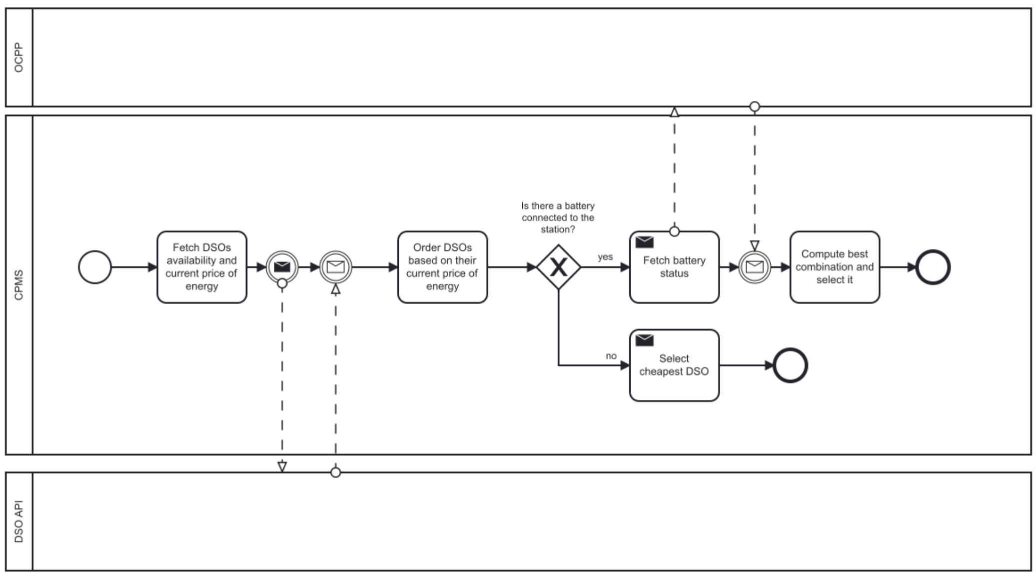

Figure 10: CPMS Dynamic selection

2.2.4 CPOW functions

- **Manual source of energy selection:** the logged CPOW visits the CPMS webapp and selects the current source of energy (or a mix between the internal battery and the external energy provider).

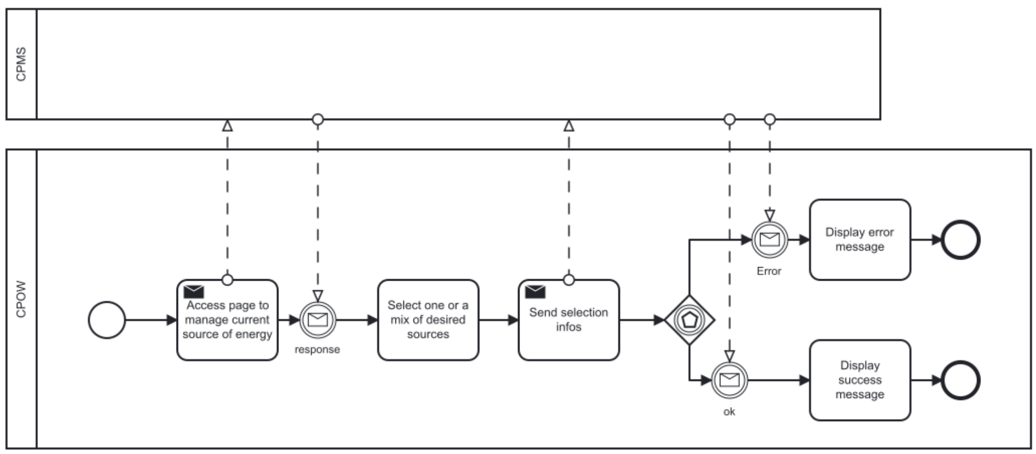

Figure 11: CPOW manual selection of the source of energy

- **Management of price and special offers:** the logged CPOW visits the CPMS webapp and can decide to change the price or to create, remove or to apply a special offer.

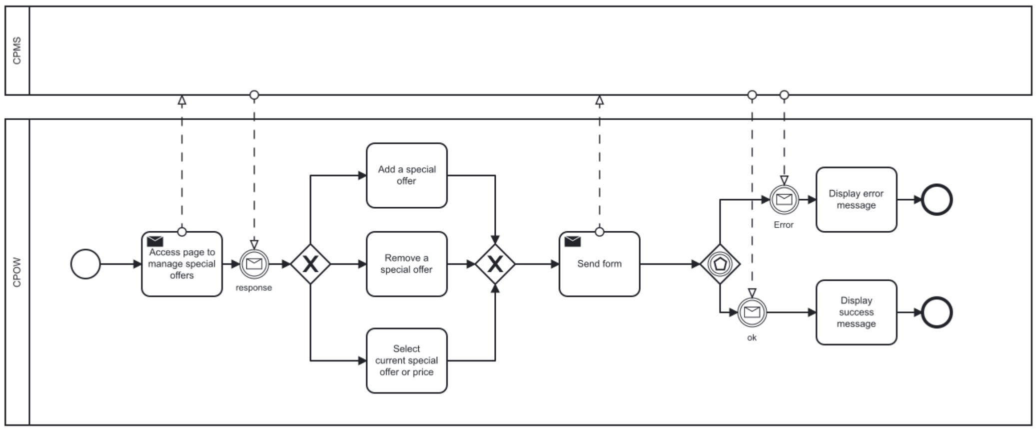

Figure 12: CPOW management of price and special offers

2.3 User characteristics

2.3.1 Electric car owner

The application has been thought for anybody that owns an electric car and wants to plan his charging process. Because of this, the potential user base comprises of any electric vehicle owner with a device connected to the internet. The user should also be capable of interacting with a webapp.

2.3.2 CPO's Human Operators (CPOW)

The CPOW is the person that manages the charging stations owned by the CPO.
He is a spacialized worker that has to be able to interact with the webapp and the CPMS in order to change from which DSO or battery the charging stations are getting the energy from, or to set special offers for the charging stations. He needs a device connected to the internet, a web browser and the company credentials to access the webapp.

2.4 Assumptions, dependencies and constraints

- D1: Users have access to the internet while using the web application
- D2: The infrastructure of the charging stations is reliable and works 99% of the time
- D3: User's car works properly while charging
- D4: Information about the charging stations offered by the CPMS are accurate (such as position, state and availability)
- D5: The DSO's infrastructure is working and their data is reliable
- D6: Information about the user's schedule is correct and meaningful
- D7: Users own an IT device to connect to the application
- D8: Each user registers only once, and always feeds correct information to the app
- D9: The data supplied by the user's devices are correct (battery charge, current position)

- D10: Special offers regarding the charging stations are correct and reliable
- D11: Special offers provided by one CPO concern all the charging stations owned by that CPO
- D12: The eMSPs and all the connected CPMSs follow the OCPI 2.2.1 protocol

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Users should interface the web application through devices that must be connected to the Internet. The service will be accessed through a web browser from the web site domain (i.g. www.eMall.com). As the service will be used by a lot of different kinds of users, the web application interface should be as simple and intuitive as possible.

3.1.2 Hardware Interfaces

Since the system provides a web application, there's not an hardware interface. In order to give a complete experience, the system should be given the access/permission to the user's GPS position or navigation system.

3.1.3 Software Interfaces

To access the application a web browser is required. Additional software like a calendar and GPS is necessary for accessing all the funcionalities of the application. The software will also take advantage of some interfaces to accomplish its requirements. An external API is necessary to visualize on maps the user location and the near charging stations.

3.1.4 Communication Interfaces

Internet connections will be mandatory since the web application must communicate with the web server and the system APIs.
Communication between the EMSP and the CPMS will be done through the OCPI protocol that allows the exchange of data in real time (RTC) and supports both peer-to-peer (P2P) and roaming hub connections. In P2P roaming, CPOs and EMSPs have direct bilateral connections via which they exchange data. Meanwhile, in hub-based roaming, a charge point operator or mobility service provider can access many roaming partners via a single, standardised connection.

3.2 Functional Requirements

3.2.1 List of Requirements

Shared Requirements

ID	Requirement
R1	The system must allow registered and logged-in users to use the app.
R2	The system must respect the GDPR and the user's privacy.
R3	The system must allow registered users to log-in using their e-mail
R4	Both the CMPS and the eMSP subsystem must abide to the OCPI 2.2.1 protocol

EMSP Subsystem Requirements

ID	Requirement
R5	The eMSP subsystem must show the user the nearby charging stations through an interactive map.
R6	The eMSP subsystem must be allowed to use the user's GPS location in order to view the nearby charging stations, if given permission.
R7	The eMSP subsystem must notify the user when the charge has finished.
R8	The eMSP subsystem must noitify the user about discounted prices.
R9	The eMSP subsystem must show the user the prices of the charging stations.
R10	The eMSP subsystem must access the User calendar schedule in order to suggest the best charging time-frames, if given permission.
R11	The eMSP subsystem must suggest the user on which charging stations to go based on the car battery and his daily schedule.

R12	The eMSP subsystem must communicate with the CPMSs in order to retrieve all needed information about the charging stations.
R13	The eMSP subsystem must allow the user to book a charging spot for a future date.
R14	The eMSP subsystem must allow the user to cancel a future reservation for a charging spot.
R15	The eMSP subsystem must allow the user to pay for the charge through an external payment service.
R16	The eMSP subsystem must allow the user to start or stop the charge via the application.
R17	The eMSP subsystem must notify the user of the upcoming booked sessions.

CPMS Subsystem Requirements

ID	**Requirement**
R18	The CPMS subsystem must communicate with the DSO according to a standard protocol.
R19	The CPMS subsystem must retrieve the DSO current energy price.
R20	The CPMS subsystem must automatically decide from which DSO to acquire energy.
R21	The CPMS subsystem must retrieve the charging station's battery status.
R22	The CPMS subsystem must dynamically change the energy source depending on the internal station's battery status and the available DSOs' prices.
R23	The CPMS subsystem must communicate the location of the charging stations to all connected eMSPs.
R24	The CPMS subsystem must retrieve the internal status of the sockets through the OCPP standard protocol.

R25	The CPMS subsystem must retrieve and communicate to all connected eMSPs the external status of the sockets through the OCPP and the OCPI standard protocols.
R26	The CPMS subsystem must allow to start or stop a charging session through the OCPP standard protocol.
R27	The CPMS subsystem must retrieve the battery status of the car through the DIN/ISO specifications.
R28	The CPMS subsystem must allow the CPOW to change the price of a charging station.
R29	The CPMS subsystem must allow the CPOW to add or change the current special offer.
R30	The CPMS subsystem must allow the CPOW to change the energy provider of a charging station.
R31	The CPMS subsystem must unlock the reserved charging spot if the user doesn't show up.

3.2.2 Mapping

User Goals

G1	Know about the charging stations nearby, including their cost and any special offer they have.
R1	Text.
R2	Text.
R3	Text.
R5	The eMSP subsystem must show the user the nearby charging stations through an interactive map.
R6	The eMSP subsystem must be allowed to use the user's GPS location in order to view the nearby charging stations, if given permission.
R8	The eMSP subsystem must noitify the user about discounted prices.
R9	The eMSP subsystem must show the user the prices of the charging stations.
D	.

G2	Book a charge in a specific charging station for a certain timeframe.
R1	Text.
R2	Text.
R3	Text.
R13	TEXT
R14	TEXT
R17	TEXT
R31	TEXT
D	.

G3	Start the charging process at a certain station.
R1	Text.
R2	Text.
R3	Text.
R4	TEXT
R7	TEXT
R16	TEXT
R26	TEXT
D	.

G4	Pay for the obtained service.
R1	Text.
R2	Text.
R3	Text.
R15	TEXT
D	.

eMSP Subsystem Goals

G5	Be able to communicate with multiple CPMS.
R4	TEXT
D	.

G6	Suggest the user convenient charging timeframes based on his upcoming schedule.
R10	TEXT
R11	TEXT
D	.

G7	Notify the user when the charging process is finished.
R7	TEXT
D	.

G8	Be able to communicate with multiple eMSPs.
R4	TEXT
R23	TEXT
R25	TEXT
D	.

G9	Be able to communicate with at least one DSO.
R18	TEXT
D	.

G10	Communicate the location and the "external" status of the charging station.
R23	TEXT
R25	TEXT
D	.

G11	Know the internal status of a charging station.
R21	TEXT
R24	TEXT
D	.

G12	Start charging a vehicle according to the amount of power supplied by the socket, and monitor the charging process to infer when the battery is full.
R26	TEXT
R27	TEXT
D	.

G13	Acquire from the DSOs information about the current price of energy.
R19	TEXT
D	.

G14	Decide from which DSO to acquire energy (if more than one is available).
R20	TEXT
R22	TEXT
R30	TEXT
D	.

G15	Dynamically decide where to get energy for charging (station battery, DSO, or a mix according to availability and cost).
R21	TEXT
R22	TEXT
R30	TEXT
D	.

CPOW goals

G16	Change the source of energy between batteries and grid.
R1	Text.
R2	Text.
R3	Text.
R30	TEXT
D	.

G17	Insert special and limited owners.
R1	Text.
R2	Text.
R3	Text.
R29	TEXT
D	.

G18	Change the price of a charging station.
R1	Text.
R2	Text.
R3	Text.
R28	TEXT
D	.

3.2.3 Use Cases

3.2.3.1 Use Cases Diagram

- **Policy Makers**

Figure 13: Policy Maker - Use Case Diagram

- **Farmers**

Figure 14: Farmer - Use Case Diagram

- **Agronomists**

Figure 15: Agronomist - Use Case Diagram

3.2.3.2 Use Cases Description

For more details 3.2.4.

- **Shared Use Cases**

Sign Up

Use Case	Sign Up
Actor	U
Entry condition	User wants to register in the system
Flow of events	1. text
Exit condition	User data are saved into the system and registration ends successfully
Exceptions	1. text An error message is shown and the flow of events starts again from point 3

- **U**

- **text**
- **Use Cases to requirement mapping** 0

1)Sign Up	R3) text R2) text

3.2.4 Sequence Diagrams

- **Shared Sequence Diagrams**

  Sign Up

Figure 16: Shared Sequence Diagram - Sign Up Sequence Diagram

- **Farmers**

3.3 Performance Requirements

The vast majority of the user base will be represented by electric car owners, while the number CPO's human operators will be negligible.
According to the European Environment Agency, 2021 saw a significant increase in the number of newly registered electric vehicles: electric car registrations for the year were close to 1.7 milions, with a 17.8% increase in the share in just 1 year. For Italy the trend is also positive with 32 thousand electric cars registered in 2021, and 67 thousand in 2020.
This means that we are expected to have thousands of registrations in the first year of the system, and a significant increase in the following years that could reach milions of users.
For this reason we should consider as our mid-term goal to have a scalable, reliable and efficient system to guarantee a good user experience.
The workload is also expected to be high so we need to ensure that the system can handle all the simultaneous requests with a good enough response time (less than 600ms should be enough).

3.3.1 Design Constraints

3.3.2 Standards compliance

eMall's user data must be compliant with the European Union's General Data Protection Regulation (GDPR).
The system must be able to store and process user data in a secure way, and must be able to provide the user with a way to access, modify or delete their data at any time. It should also ask the user for the permission to activate the GPS tracking feature and the permission to access the user's calendar to schedule the charging sessions, and must be able to revoke these permissions at any time.

3.3.3 Hardware limitations

The software must be able to run on any device with a modern web browser (Chrome, Firefox, Edge, Safari, Opera, etc.).
Given the geographical context of the project, the main limitations are related to the internet connection speed.
Also, in order to fully use the system, the user must have a smartphone with a GPS sensor and a calendar app installed.

3.4 Software System Attributes

3.4.1 Reliability

The system must have a backup infrastructure to ensure that the data is always available and that the system can be restored in case of a failure.
It must be characterized by an high MTTF (Mean Time To Failure) and a low MTTR (Mean Time To Repair).
All the needed components must be redundant and parallelized as much as possibile to ensure an high reliablity of the overall system in case of a single component failure (as shown in Fig. 17).

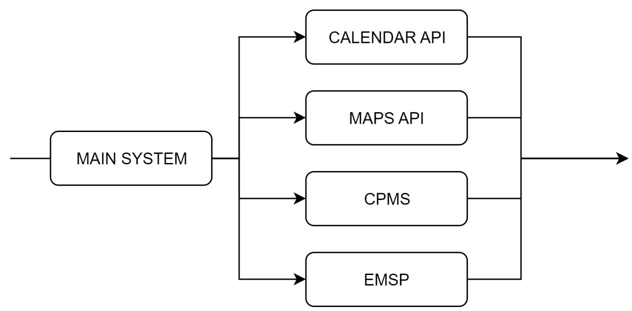

Figure 17: eMall - Components

3.4.2 Availability

The system should be accessible 24/7, with a downtime of less than 1 hour per year. The peak hours are expected to be during the working hours, especially during lunch-breaks, so the system should be able to handle the workload during these hours. For this reason it would be better to do all the manteinance activities during the night, when the system is not used.

Component	Availability	Downtime
Main System	99.99%	52 min/y
Calendar API	99.999%	5 min/y
GPS API	99.999%	5 min/y
CPMS	99.99%	52 min/y
EMSP	99.99%	52 min/y
Entire System	99.99%	52 min/y

3.4.3 Security

All the user data must be stored in a secure way, and must be hashed and encrypted using approved hash algorithms like SHA-256 and strong encryption algorithm like AES-256. The communication between the client and the server must be encrypted using TLS 1.3 and powered HTTPS. All stored data must be compliant with the GDPR.

3.4.4 Maintainability

The system must be easy to maintain and to update, and must be able to be extended with new features in the future. For this reason the system must be designed with a modular architecture, and must be developed using modern design patterns and best practices to guarantee high reusability.
The code should also be well documented and easy to understand, and must be tested using unit tests and integration tests.

3.4.5 Portability

The system will be designed to work with most of the modern web browsers (Chrome, Firefox, Edge, Safari, Opera, and possible other Chromium based web browsers) and most of the modern operating systems (Windows, Linux, MacOS, Android, iOS).

4 Formal Analysis using Alloy

4.1 Formal Analysis Purpose

The following analysis aims to formally prove the correctness of the system model by exploiting Alloy verification tool. To achieve so, we test the model by checking if some of the previous defined goals are met.

Specifically, we stress:

- **(G3)** :

4.2 Alloy Code

```
module projectSE2
open util/boolean

abstract sig User{
        device: one SmartDevice,
        email : one Email,
        password : one string,
        name : one string,
        surname: one string
}

sig CarOwner extends User{
        car : set Car,
        schedule : one DailySchedule,
        suggestedChargingTimes: set Event,
        suggestedChargingLocations : set Location,
        bookings : set Booking,
        nextTrip: lone Route,
        nearStations: set ChargingStation
}

sig CPOoperator extends User{
        company : one string
}

sig Offer{
        author : one CPOoperator,
        value : one Value,
        startingDate: one Date,
        endingDate: one Date,
}{
        startingDate ≠ endingDate
}

sig Value{
        tens : one Int,
        units : one Int
}{
```

```
        tens≥0
        units ≥0
        tens + units > 0 //offer cannot be 0% discount
}

sig ChargingStation{
        owner : one string,
        sockets : disj some Socket,
        totalSockets: one Int,
        location : one Location,
}{
        totalSockets = #sockets
        totalSockets>0
        }

sig SmartDevice{
        currentLocation : one Location
}

sig Location{
        coordinates : one Coordinates
}

sig Coordinates{
        longitude : one Int,
        latitude : one Int
}

sig Route{
        stops: some Location
}

sig Email{}
sig string{}

sig Socket{
        chargingSpeed : one Int,
        occupied: one Bool,
        bookings : Booking,
        socketType : one string
}

sig Car{
        batteryPercent: one Int,
        maker : one string
}

sig Booking{
        location : one Location,
        time : one Date,
        socket : one Socket,
        user : one Email
}

sig DailySchedule{
        events : set Event
}
```

```
sig Event{
	location : one Location,
	time : one Date
}

sig Date{}

sig CPMS{
	CPOowner : one string,
	operators : set CPOoperator,
	chargingStations : disj some ChargingStation,
	allOffers : disj set Offer,
	offers: allOffers-> some chargingStations
}{
	#operators>0
}

one sig System{
	cpms : some CPMS,
	users : some User
}

//a socket is only in one charging station
fact socketHasOneChargingStation{
	all s: Socket | no disj cs1, cs2 : ChargingStation | s in cs1.
	    ↪ sockets and s in cs2.sockets
}

//CPO operators are registered only in CPMSs owned by their company
fact correctOperators{
	all cpow : CPOoperator, cpms : CPMS | cpow in cpms.operators iff
	    ↪ cpow.company = cpms.CPOowner
}

//there are no sockets without charging stations
fact socketIsInCS{
	all s: Socket | one cs : ChargingStation | s in cs.sockets
}

fact nearbyStationsConditions{
	all cs:ChargingStation, u:CarOwner | cs in u.nearStations iff cs.
	    ↪ location=u.device.currentLocation
}

//all charging stations are owned by a CPO
fact noWildChargingStation{
	all cs : ChargingStation | one cpms: CPMS | cs in cpms.
	    ↪ chargingStations
}

//no uregistered users or CPMSs
fact allUsersRegistered{
	all u: User | one s: System | u in s.users}

fact allCPMSInSystem{
	all ms:CPMS | one s:System | ms in s.cpms}
```

```
//charging stations in cpms and the cpms itself have the same CPO owner
fact sameOwner{
all cpms: CPMS | all cs : ChargingStation | cs in cpms.chargingStations iff
    ↪ cs.owner=cpms.CPOowner
}

//no car has 2 different owners
fact noCarHasTwoOwners{
        all c : Car | no disj u1, u2 : CarOwner | c in u1.car and c in u2.
            ↪ car
}

//all cars have a registered user
fact allCarsHaveOwners{
        all c: Car | one u: CarOwner | c in u.car
}

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

fact allOperatorsWorkForCompanies{
        all o:CPOoperator, cpms:CPMS | o.company = cpms.CPOowner implies o
            ↪ in cpms.operators
}

//all bookings belongs to users
fact trueBookings{
        all b: Booking | all u : User | b.user=u.email iff b in u.bookings
}

//no different sockets have same booking
fact {
        no disj s,s1:Socket | some b:Booking | b in s.bookings and b in s1.
            ↪ bookings
}

//mapping booking and sockets
fact BookingToSocketRelation{
        no b: Booking, s: Socket | b in s.bookings and b.socket ≠ s
}

//mapping email and users
fact noWildEmails{
        all e: Email | one u: User | e = u.email
}

//no shared schedules
fact DifferentSchedules{
        no disj c1,c2: CarOwner| c1.schedule = c2.schedule
}

//no shared devices
fact noSameDevice{
        no disj  c1,c2: CarOwner| c1.device = c2.device
}
```

```
// all charging station locations are distinct
fact noSameLocationForCS{
        no disj cs1, cs2: ChargingStation | cs1.location = cs2.location
}

//when the reservation is stored, location in the reservation should match
    ↪ the location of the charging station booked
fact BookingRightLocation{
        all b: Booking, cs:ChargingStation | b in  cs.sockets.bookings iff b
            ↪ .location=cs.location
}

//no disjointed locations have the same coordinates
fact allLocationsAreDifferent{
        no disj l1,l2 : Location | l1.coordinates=l2.coordinates and
                l1.coordinates.latitude=l2.coordinates.latitude and
                l1.coordinates.longitude=l2.coordinates.longitude
}

fact allOffersAreLegitimate{
        all cpms:CPMS, offer: Offer | offer in cpms.allOffers iff offer.
            ↪ author in cpms.operators
}

fact noOverlappingEventsForUser{
        all ds: DailySchedule | no disj e1, e2: Event | e1 in ds.events and
            ↪ e2 in ds.events and e1.time = e2.time
}

fact SuggestCorrectly{
        all c: CarOwner, e: Event | one cs :ChargingStation | e.location=cs.
            ↪ location
                and e in c.schedule.events
                        and e in c.suggestedChargingTimes
}

fact SuggestLocationsCorrectly{
        all c:CarOwner, place:Location | one cs:ChargingStation |
        place in c.suggestedChargingLocations iff (place in c.nextTrip.stops
            ↪  and place = cs.location)
}

fact noWildSchedules{
        all s:DailySchedule | one sys:System | s in sys.users.schedule
}

fact noWildDevices{
        all d:SmartDevice | one sys:System | d in sys.users.device
}

fact noWildLocations{
        all l:Location | one sys:System | l in sys.users.device.
            ↪ currentLocation or l in sys.users.schedule.events.location or
            ↪  l in sys.cpms.chargingStations.location
}
```

```
fact allBookingsAreRegistered{
        all b:Booking | one cs:ChargingStation| b.location=cs.location and b
            ↪  in cs.sockets.bookings
}


assert OfferValidity{
        all offer:Offer, cpms:CPMS, cs:ChargingStation, sys:System |
        (offer->cs) in cpms.offers implies
                (offer.author in sys.users
                and offer.author.company=cpms.CPOowner
                and cpms.CPOowner = cs.owner and offer in cpms.allOffers
                and cs in cpms.chargingStations)
}

//verify that all the bookings made are registered to the right user, and
    ↪ location and socket of the charging station matches correctly
assert BookingsAreRegistered{
        all b: Booking, cs : ChargingStation, c : CarOwner |
                        (b in c.bookings iff
                                                b.user=c.email)
                        and (b in cs.sockets.bookings  iff
                                                b.location = cs.location
                                                and b.socket in cs.sockets )
}

assert CorrectNearby{
 all disj co:CarOwner, cs:ChargingStation | cs in ChargingStationIsNearby[co
     ↪ , cs] iff co.device.location=cs.location
}

//verify that only correct suggestions are made
assert SuggestionWorksCorrectly{
        all c: CarOwner, e: Event | some cs:ChargingStation |
                e in c.suggestedChargingTimes iff
                e.location in cs.location
                and e in c.schedule.events
}

assert LocationSuggestionWorksCorrectly{
        all c:CarOwner, place:Location | some cs: ChargingStation|
        place in c.suggestedChargingLocations iff
        (
                place in c.nextTrip.stops and cs.location=place
        )
}

fun ChargingStationIsNearby[co:CarOwner, cs:ChargingStation]: lone
    ↪ ChargingStation{
        {return: ChargingStation | (return=cs iff co.device.location=cs.
            ↪ location)
        }
}

pred bookingRequest[b:Booking, co:CarOwner, cs:ChargingStation]{
        (b in co.bookings) implies
                (b.user=co.email)
```

```
        and (b in cs.sockets.bookings) implies
                (b.location = cs.location
                and b.socket in cs.sockets)
}

pred ChargingProcess[co: CarOwner, cs: ChargingStation, s: Socket, b:Booking
    ↪ ]{
        co.device.currentLocation = cs.location and s in cs.sockets and s.
            ↪ occupied.isTrue and b in co.bookings and b in s.bookings
}

pred NearbyStation[u:CarOwner]{
        #u.nearStations>0
}

check OfferValidity for 10
check CorrectNearby for 10
check SuggestionWorksCorrectly for 10
check LocationSuggestionWorksCorrectly for 10
check BookingsAreRegistered for 10

pred show{

}

run NearbyStation for 10
run ChargingProcess for 10
run bookingRequest for 10


run show{
#offers>1
#CPOoperator>3
#CPMS > 1
#Offer > 1
#CarOwner > 1
#ChargingStation >1
#Socket>2
} for 15
```

Plus, we runned some empty predicates `show` to generate some more general instances of the app system.