AY 2022/2023

Politecnico di Milano

RASD: Requirement Analysis and Specification Document

Valeria Amato Francesco Dettori Matteo Pancini

Prof.
Elisabetta Di Nitto

Version 1.4
December 23, 2022

Contents

1 Introduction **1**
- 1.a Purpose . . . 1
- 1.b Scope . . . 2
  - 1.b.1 Description of the given problem . . . 2
  - 1.b.2 Phenomena . . . 2
- 1.c Definitions, Acronyms, Abbreviations . . . 4
  - 1.c.1 Definitions . . . 4
  - 1.c.2 Acronyms . . . 4
  - 1.c.3 Abbreviations . . . 4
- 1.d Revision History . . . 5
- 1.e Reference Documents . . . 5
- 1.f Document Structure . . . 6

2 Overall Description **7**
- 2.a Product Perspective . . . 7
  - 2.a.1 Scenarios . . . 7
  - 2.a.2 Class diagrams . . . 10
- 2.b Product Functions . . . 11
- 2.c User Characteristics . . . 13
- 2.d Assumptions, dependencies and constraints . . . 13

3 Specific Requirements **15**
- 3.a External Interface Requirements . . . 15
  - 3.a.1 User Interfaces . . . 15
  - 3.a.2 Hardware Interfaces . . . 20
  - 3.a.3 Software Interfaces . . . 21
  - 3.a.4 Communication Interfaces . . . 21
- 3.b Functional Requirements . . . 21
  - 3.b.1 Requirements . . . 21
  - 3.b.2 User Use Case Scenarios . . . 22
  - 3.b.3 CPO Use Case Scenarios . . . 34
- 3.c Requirements Mapping . . . 42
- 3.d Design Constraints . . . 47
  - 3.d.1 Hardware Constraints . . . 47
  - 3.d.2 Privacy Constraints . . . 48

4 Software System Attributes **49**

5 Formal Analysis using Alloy **50**
5.a Model 50
5.b Worlds 61

6 Effort Spent **65**

1 Introduction

eMall (e-Mobility for All) is an interactive and easy-to-use application whose aim is to limit the carbon footprint caused by our urban and suburban mobility needs. One of the main problems with electric vehicles is the constant worry of the charging process planning. In this direction eMall wants to be fully integrated into everyone's daily schedule, in order to let the user charge the vehicle and carefully plan the charging process without interfering with their daily activities. eMall is a cutting-edge approach to a brighter, more environmentally friendly future.

1.a Purpose

This document represents the Requirement Analysis and Specification Document (RASD) for eMall. The aim is to outline both functional and non-functional requirements for the system's development, examine use cases, features, and user interaction, and highlight the restrictions and limitations of the software that will arise once it is made available. This document is addressed to the developers who have to implement the requirements and may be used as an agreement between the client and the contractors.

The main objectives of our system are:

- G1: Allow users to visualize chargers nearby, their cost and special offers.
- G2: Allow users to book a reservation for a charge at a specific location and at a specific time and timeframe.
- G3: Allow users to start the charging process
- G4: Let the system notify the user when the charging process is over
- G5: Allow users to pay for the service
- G6: Let the system notify the user with calendar charging suggestions
- G7: Allow the CPOs to globally manage their charge stations
- G8: Allow the CPOs to manage charging points
- G9: Let the system select on its own which DSO is the best to obtain energy from due to energy availability and cost

1.b Scope

Electric vehicles use electricity to charge their batteries instead of using fossil fuels like petrol or diesel. Electric vehicles are more efficient, and that combined with the electricity cost means that charging an electric vehicle is cheaper than filling petrol or diesel for your travel requirements. Due to the longer charging times and lower number of charging stations than petrol stations, electric mobility does require more planning. The company eMall is working to develop a solution so that an electric car owner won't even notice the disadvantages related to charging time of this sort of vehicle, creating a platform which is perfectly integrated in the owner's daily schedule. Moreover, eMall wants to help CPOs manage their stations, let them know information about the internal and external situation, deal with DSOs to acquire energy and force the start and stop of charge in case of need.

1.b.1 Description of the given problem

Electric vehicles use electricity to charge their batteries instead of using fossil fuels like petrol or diesel. Electric vehicles are more efficient, and that combined with the electricity cost means that charging an electric vehicle is cheaper than filling petrol or diesel for your travel requirements. Due to the longer charging times and lower number of charging stations than petrol stations, electric mobility does require more planning. eMall is working to develop a solution so that an electric car owner won't even notice the disadvantages related to charging time of this sort of vehicle, creating a platform which is perfectly integrated in the owner's daily schedule.

1.b.2 Phenomena

According to the paper "The World and the Machine" by M.Jackson and P.Zave, we can identify the application domains. In the following table are discussed the world phenomena, the shared phenomena and the machine phenomena. For the shared phenomena we will describe whether they are controlled by the world or the machine.

World Phenomena

- WP1: People need to charge their car

- WP2: E-vehicles need to be charged
- WP3: An unregistered user checks his email to validate his account
- WP4: An unregistered user validates his account through the link sent by email
- WP5: A CPO needs to acquire energy for his stations
- WP6: A CPO needs to know the battery status of his station

Shared Phenomena

- SP1: A user books a charging reservation (Machine)
- SP2: A user selects day/time/timeframe of reservation (World)
- SP3: An unregistered user fills up the registration form (World)
- SP4: The system sends a confirmation email to the user to validate its account through a link (Machine)
- SP5: An unregistered user signs up (World)
- SP6: A registered user logs in (World)
- SP7: The system updates charging station data (Machine)
- SP8: The user enters his payment method (World)
- SP9: The user gives permissions to use his GPS location (World)
- SP10: A CPO manages his stations (World)
- SP11: A logged-in user logs out (World)

Machine Phenomena

- MP1: The system sends a notification
- MP2: The system create a reservation suggestion
- MP3: The system generate a unique ticket id

1.c Definitions, Acronyms, Abbreviations

1.c.1 Definitions

- Customers: the people whom this service is directed to. They can belong to any gender but they must have a driver license. Their main purpose is to request a ticket to schedule their line up at the shop.
- User: it is a customer who registers and then uses the application.
- Actor: identifies a user or a CPO.
- Charging station: set of one or more chargers in which users can charge-up their e-vehicles.
- Ticket: pass generated by the system which contains a unique ID to unlock the charging process

1.c.2 Acronyms

- CPO: Charging Point Operator. It refers to the charging station owner and manager.
- CPMS: Charge Point Management System. System through which each CPO can administer his own IT infrastructure.
- DSO: Distribution System Operator. It refers to 3rd party companies that sell energy to CPOs.

1.c.3 Abbreviations

- UCn: use case number n
- Gn: goal number n
- Rn: requirement number n
- DAn: domain assumption number n
- ID: identifier. Ir is a unique sequence of numbers or letters in order to unambiguously identify an entity.

1.d Revision History

- November 30, 2022 (version 1.0)
- December 9, 2022 (version 1.1):
  - Update of the Reference Documents
  - General revision
- December 19, 2022 (version 1.2):
  - Effort spent
  - General revision
  - Update worlds
  - Update sequence diagrams
- December 21, 2022 (version 1.3):
  - Crosscheck with DD
  - General revision
- December 24, 2022 (version 1.4):
  - Typo corrections

1.e Reference Documents

- Specification document: R&DD Assignment - A.Y. 2022/23
- Alloy official documentation: https://alloytools.org/documentation.html
- Course slides
- Paper: "Jackson and Zave: the world and the machine"
- UML offcial specifcation: https://www.omg.org/spec/UML/

1.f Document Structure

The document will be structured by the following sections:

- **Section 1: Introduction**
  This section will provide the purpose and scope of the system. It will show the main goals and the world and shared phenomena, as well as other information such as definitions, acronyms and abbreviations that will be used in the whole document.

- **Section 2: Overall description**
  This section will offer a summary description about the overall organization of the system. We will show the system through class diagrams, scenarios and we will explain the main hardware and software costraints.

- **Section 3: Specific Requirements**
  This section will contain several visual mockups with a brief description of the system. Morevover functional and performance requirements will be analyzed as well as design constraints and software system attributes. In order to do this we will provide use case diagrams, activity diagrams and the mapping on requirements.

- **Section 4: Formal Analysis through Alloy**
  This section will include a formal analysis of the system using alloy with its objectives, model and predicates execution. Finally some results of the worlds will be executed.

- **Section 5: Effort spent**
  It will show the time spent by each team member on this document.

2 Overall Description

2.a Product Perspective

The product we are going to develop is composed of two main parts: the application interface and the system. The app, that could be used as a mobile app (mainly for customers) and as a web app (mainly for CPOs) works as an interface while the system works as the business side of the application and it will be responsible for data management and operations functionality. The product takes advantage of some useful integrations, such as external API interfaces for some features. It also uses pre-existent graphic packages in order to make the user experience more enjoyable. Even if eMall provides all the modules that are necessary for its execution, it must exploit external interfaces to accomplish its requirements.

2.a.1 Scenarios

- **Scenario 1**
  Attila is a boy from Sardinia who drives an electric car. The car's battery is almost dead and he needs to get to a business appointment in a nearby town. In Sardinia, there aren't many charging stations, so Attila has to go to outlying villages in search of one. With very little remaining range, the guy arrives at a charging station, but all of the stations are occupied, so he has to wait for two hours for one to become free. When it is finally his turn, he realizes he missed the meeting and decides to go to a bar to kill time. Uncertain of when the battery would have recharged, he has to get up from the table numerous times to check. He chooses to pay at the station management when the recharge is finally concluded, but the manager has a temporally malfunctioning POS and only accepts cash. A framework for managing charging stations called eMall is introduced later. Attila instantly installs the eMall app after learning about it. He has a meeting today from 3 to 6 p.m. in a nearby city, and his car has a limited range. Having signed up, logged in, and connected his car and his credit card account to the app, he chooses to schedule a recharge. He chooses a station from the map that is conveniently close to the meeting place, reserves it for the three hours he will be busy, and as soon as he receives notification that the refill is finished, he pays automatically for

the service. No more missing business meetings!

- **Scenario 2**
  Aristide is a middle-aged man who lives in a town near Turin. Since he purchased his first electric vehicle, he has the eMall app installed on his phone. When Aristide needs to recharge his vehicle, he opens the app, and begins looking for charging stations nearby. He may do this using the search bar or the map, and he can set filters for things like distance, cost, type of columns (slow, fast, rapid), green energy, availability, and special deals. Aristide picks a charging station and specifies the time interval for which he wants to reserve it, the type of column and the energy mix, creating a reservation ticket. But as soon as he creates the ticket, he glances at the weather report and sees that a thunderstorm is predicted. He then cancels the original ticket and generates a new one for the following day. A few days later, when he connects to the reserved column and clicks the "Start" button on the ticket within the app, charging begins immediately, and an alert is sent to show the status has changed. Aristide can go back to his house in the interim and watch his preferred program. Aristide can walk over to the car to pick it up once another notification that the recharge is complete has been received. He will receive notification that automatic payment has been made for the service he recently used as soon as he unplugs the column.

- **Scenario 3**
  Fisietto is an elderly gentleman living in Paris. He used the eMall app to reserve a charge before departing the house, leaving his phone on the couch. At the appointed time, he arrives at the charging station, walks up to the column, and enters his license plate. Fisietto returns home once the charging is finished and checks his smartphone to find the payment notification from eMall congratulating him for using the service.

- **Scenario 4**
  TeslEl, a sizable Milan-based corporation well known for creating cutting-edge electric cars, is led by Lazzarella. Lazzarella has chosen to participate with the eMall initiative because she firmly believes in it. Since Lazzarella has a hectic schedule, eMall proposes the most convenient

times of day to charge her TeslEl. TeslEl's battery is continually being monitored by eMall, and when a recharge is necessary, it searches nearby stations for deals. It notifies her with a proposal for a reservation when it locates the best station with free charging sockets at the most practical time of day for Lazzarella. She then schedules a recharge for that period, however she gets an unexpected call as the recharge is being completed, so Lazzarella cancels it and reschedules it for another convenient day.

- **Scenario 5**
  Ludmilla is a young CPO residing in Stockholm. She leaves her home in the cold every morning, and throughout the day, she is required to periodically check the external and internal conditions of the charging station to which she is assigned. She has to sprint to the customer as soon as he arrives, start the machine's charging procedure, and keep an eye on it until it is finished. She also has to visit the websites of the numerous DSOs and choose the one that is most convenient to her. Ludmilla returns home exhausted and with a cold. A few days later, eMall is launched, allowing Ludmilla to work comfortably from her office while sipping hot tea. She launches the eMall Web App and logs into the service. She can control the entire charging station from here, keep an eye on the charging process, get data from DSOs, and establish pricing and promotional deals.

- **Scenario 6**
  Luismarida is a CPO living in Naples. Luismarida launches the eMall Web App and logs in as soon as she gets to the office on her first day of work. She learns that she may check the charging station's internal and external status via the dashboard, including: the number of charging outlets that are available, their type, their price, the energy mix, and, if any are reserved, an estimation of how long it will take for one to become free. Luismarida may also alter prices and energy mix, provide promotional offers, show how much energy is left in the batteries, as well as how much energy has been used up so far by each car being charged and how long it will take to finish charging. Being able to complete her task conveniently and safely makes Luismarida ecstatic.

- **Scenario 7**
  Theodosius is a CPO living in London. A customer shows up at the

charging station on a nasty rainy day. Theodosius is delighted that he can now monitor the charging process from the eMall Web App, leaving it up to the system to start the charging process and determine when the car battery is charged, instead of having to endure a rain. Additionally, Theodosius has the ability to cause the charging process to begin and end.

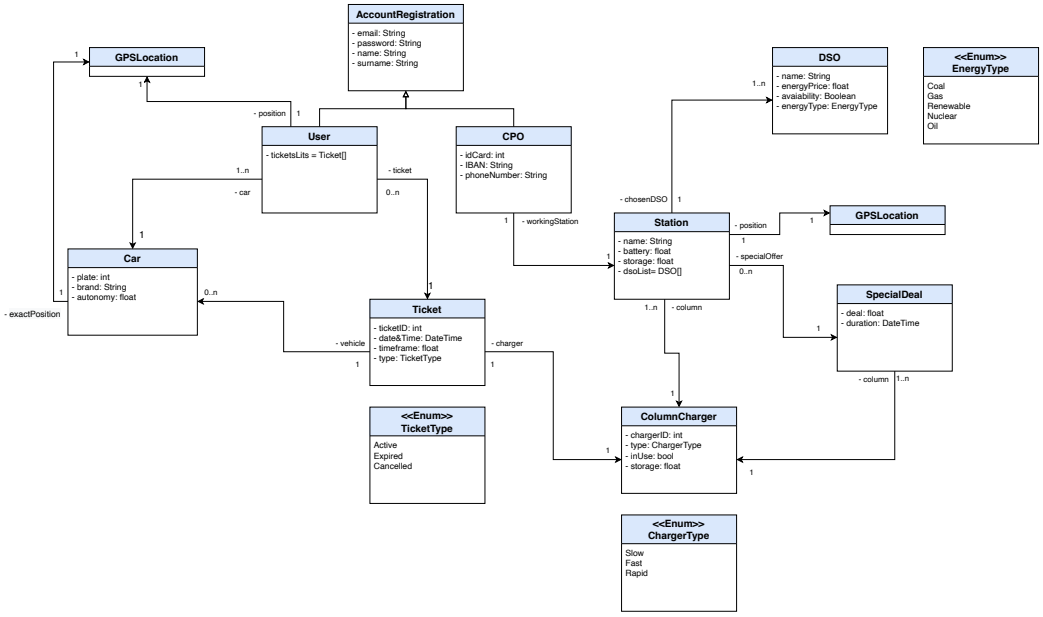
- **Scenario 8**
  Arruda is a CPO living in Lugano. A friend asks her how it is possible that her charging stations always have the most competitive prices. Arruda, proud, explains to her that she uses eMall and that thanks to its Web App she is able to acquire information on the price of energy offered by the various DSOs and to select the cheapest offer, based on price and energy mix used. Finally, to further optimize the pricing process it manages the distribution of energy, deciding whether to take it from DSOs, station batteries or a mix of them. The friend looks at her with a look of admiration.

2.a.2 Class diagrams

The class diagram below is a high-level representation of the whole system. The main elements are:

- **AccountRegistration**: identifies two categories of users which can access the system with credentials: CPOs and Users. This distinction is due to different registration processes and credential needs.
- **User**: identifies the person registered to eMall with all his personal information.
- **CPO**: identifies a registered CPO with all his personal information. In addition, a CPO must have registered at least one station to use the service.
- **Ticket**: identifies the reservation ticket for a single charging process. A ticket represents the only way to unlock a charger using the eMall platform. A ticket could be used by inserting into the charger either its ID or the vehicle plate of the car related to the ticket.

- **Car**: identifies the car owned by the user. The main attribute plate is an important feature because it is part of the ticket, and permits to use the service even in cases in which the mobile application doesn't work or the phone runs out of battery.
- **Charging Station**: identifies the CPO's station, with all the charges which the station is composed of.
- **Charger**: identifies a single charger, with all the features the define its type, its state of use and its charging parameters.



2.b Product Functions

- **Sign Up**
  This functionality lets the customer register in order to use eMall. User will be able to register with username and password or via social accounts. If the former is chosen, after inserting his credentials, the user will receive a confirmation email in order to validate his account. Then, the user is taken to the home page. On the other hand, by choosing a social account registration (Facebook or Google), the user will be firstly redirected to the login page of the social and then taken to the homepage of the app. After the email check the user is asked to insert

his vehicle information and a payment method in order to access the app.

- **Book a Ticket**
  This functionality is available to all users. The booking could be activated in 2 different ways: From the "map page" the user clicks on the station desired. A pop-up appears in the interface with the information to create a ticket: charger available (with relative price), date, time and timeframe. For each charger the user can select the date, time and timeframe available. From the "calendar suggestion" that is a pop-up notification on the mobile device of the user. Once clicked on the notification, the user will be able to directly book the charging time slot.

- **Charging process**
  The user will be able to activate the charging process of the pump connected to the car through a button and he/she will be notified when the vehicle is fully charged. In addition the user is able to force the stop before the actual time.

- **Add discounts**
  The CPO can add discounts on the managed charging stations/points by clicking on the designated button on the charging station page. In this way a CPO can create special offers, for a limited amount of time, available to all the users.

- **Change the energetic mix of a charging point**
  The CPO can change the energetic mix of a charging point or of the whole charging station by clicking on the designated button of the energetic mix.

- **Start/Stop the charging process**
  The CPO is able to force the start/stop of a charging process according to the amount of power supplied by the socket, and monitor the charging process to infer when the battery is full.

- **Buy DSO energy**
  The CPO can buy different types of energy in the DSO Management section. He/She has the possibility to change the percentage of the type of energy bought.

2.c User Characteristics

- **User**
  It is a person who registered an electric vehicle on the app and uses eMall as the mail platform to charge it. It is already registered to the app so it is able to use all of its functionalities. Thanks to the app he can book a reservation for charging his car without interfering with his daily work, being sure not to get stranded out. He can also take advantage of the smart calendar suggestions that the app advises him.

- **CPO**
  It's the owner and manager of a charging station. He can know the external status of the station, so all the info related to the charging sockets, especially their status. He can also know the internal status of the station, so the amount of energy stored, consumption and charging info. He can get info about DSOs' prices of energy and, in case of scarcity, buy energy from them.

2.d Assumptions, dependencies and constraints

Below the list of the domain assumption:

- D1: The payment method must be correct
- D2: The events in the calendar must be correct
- D3: The user must enter the locations of the events in his calendar
- D4: The user must let the system to access the GPS
- D5: Each user must insert correct data as input (i.e. car information, correct data, etc.)
- D6: GPS position of the charging station is exact and leads to that charging station
- D7: All users have access to internet connection
- D8: The recommendations are based on the calendar of the user, especially on the time and position of his activities

- D9: The recommendations are such that there are not interference in the same station
- D10: The CPO must have an account

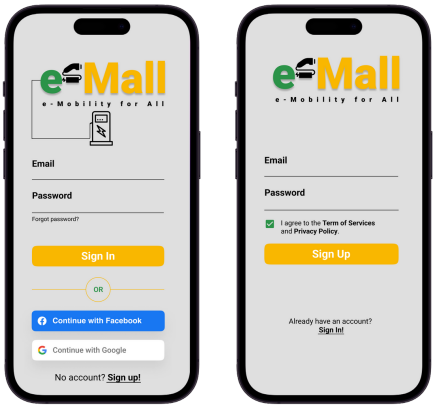
3 Specific Requirements

This part is devoted to giving a thorough explanation of each need the system must satisfy in order to carry out all of the previously described functionalities.

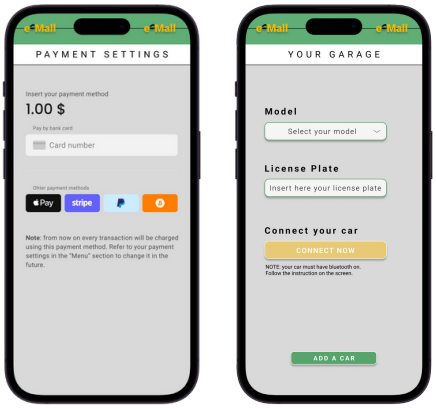
3.a External Interface Requirements

3.a.1 User Interfaces

Customer Interface The application's opening screen is depicted in the images below. The user is prompted to sign in or establish an account. The user must provide his email address and a password in order to establish a new account, or maybe choose to join up using a Google or Facebook account.



Since it is a fully digital software, after the email confirmation a user must insert his payment information, which will be used to automatically pay the charge in a contactless and integrated way. Moreover the user is asked also to add his car information. This feature can be triggered also inside the app to add more cars.



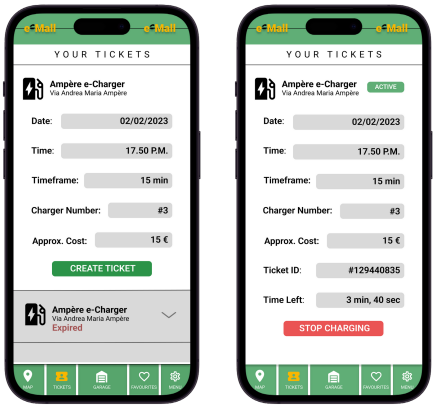
A CPO must also complete the service's registration process. Due to security and reliability concerns, the CPO setup process is more complicated. A CPO must link his account to his station (name, position, number, type of charger, etc.). To prevent incorrect registrations, the system will validate the data (such as errors in the information of the station).

The app is based on 5 main pages:

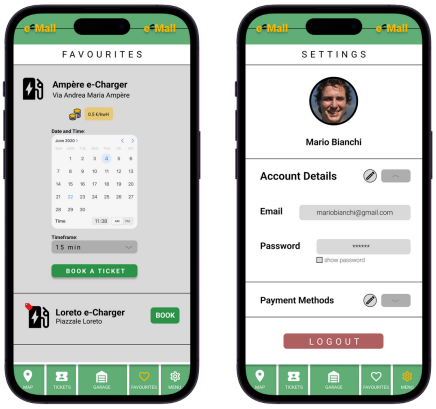
- Map page
- Ticket page
- Garage page
- Favourites page
- Menu page

The map page displays your location and the closest e-charger in real time. It enables you to search for stations and sort them (by price, distance, eco-friendliness, special offers, etc.). and ultimately click on a certain station to reserve a ticket. You can select the charging type, the associated costs, the date, the start time, and the timeframe in the ticket pop-up.

Your active and expired tickets are summarized on the ticket page. Each current ticket includes the approximate cost of the charge, the date, the time, and the duration. You can begin your charge from this interface (within the confines of your timeline). The system will now generate a ticket ID, which you'll enter on the charger's smart screen to unlock it in case the charging process doesn't start automatically through the app. The charger determines that the car's battery is full using APIs, or you can stop the charging process manually. You could also enter your license plate in place of the ticket ID so that the system will unlock your charger and enable payment without the need for the app. It is obvious that a reservation needed to be made in advance. You may also check the live battery state of your vehicles on the garage page. The software notifies the user and suggests him to purchase a ticket for a car in need of batteries, also checking the available special offers of the charging stations in the nearby. Moreover the system proposes a potential time and duration from the ticket area that properly fits your day and your car's battery needs as it is linked to your calendar (by using APIs).



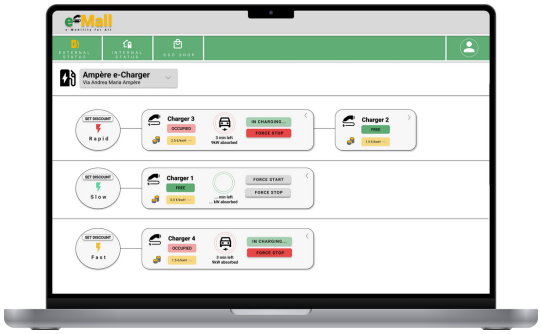
The favorites page allows you to instantly book a ticket and displays all of your favorite stations. The account information, and payment options are all available on the menu page.



CPO Interface The CPO interface is built on a web application with 3 main pages:

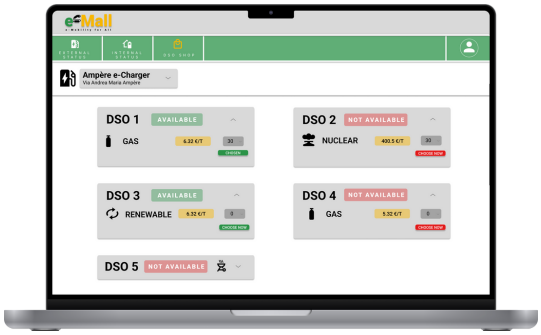
- External status page
- Internal status page
- DSO market page

Every charger is listed on the external status page with information about its type, whether it is free or occupied, and eventually the amount of kW consumed and remaining charge time. The CPO is able to configure discounts, prices for each type of charger, and the energetic mix using this interface.



An overview of the station's current condition, including information on the battery and energy mix, is provided in the internal status page.

The DSO shop page lists every DSO that is available along with energy prices and the option to purchase it.



3.a.2 Hardware Interfaces

Our project is a software-based platform, but in order to maximize its potential, it needs specific hardware interfaces. From the perspective of the user, all that is required for the system to work with the integrated GPS

module is a smartphone. The CPOs interface is a web application that can be used from any device with an internet connection. The chargers must have an inbuilt touch screen that the user may use to insert the ticket ID or the license plate to unlock it. The screen can also be used by consumers who may not have been able to book a ticket through the app in order to make a new ticket and use the charger, if free of reservations.

3.a.3 Software Interfaces

To give the user an unique and efficient service, the software needs some APIs.

The map provides the user's interface for searching for stations directly in a real-world view, and it also provides a visual representation of the distances and some basic information.

The calendar serves as the user interface for interacting with the user's schedule and providing the best options for charging the vehicle via AI algorithms.

Each charger has software that checks the ticket's validity. It eventually produces a fresh, legitimate ticket as well. In order to know when the batteries are full, it can also link to the car's battery status.

3.a.4 Communication Interfaces

The devices on which eMall is running communicate using standard Internet protocols, mostly HTTP.

3.b Functional Requirements

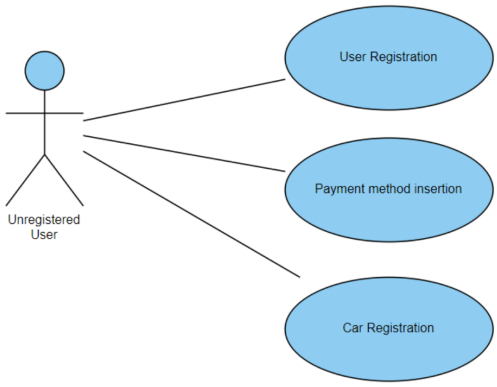
3.b.1 Requirements

- R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
- R2: The system allows the user to log in by entering the credentials used at the time of registration.
- R3: The system must regularly update charging station data regarding cost and special offers
- R4: The system allows the user to give permission about GPS location

- R5: The system shall allow users to select a day, time and timeframe slot from the available ones
- R6: The system generates a TicketID associated to a charging reservation
- R7: The system computes a prediction of expected cost of the charge, based on the timeframe and charging info and lets the user pay automatically at the end of the charge.
- R8: The system enables users to insert the license plate onto a smart pump.
- R9: The system enables users to press the "start charging" button on the application
- R10: The system allows the user to connect its own calendar to the e-mall application
- R11: The system recommends the user a ticket reservation in case it detects car's battery need by elaborating the best charging time to fit its calendar
- R12: The system allows the CPO to know main information (occupied, type of charging, cost, time remaining, energy consumed, energy mix)
- R13: The system allows the CPO to know main information related to the whole station (battery, status, number of cars in charge)
- R14: The system allows the CPO to see the price of energy from the DSO
- R15: The system allows the CPO to change the DSO provider and the energy mix for the whole station (all charging points)
- R16: The system allows the CPO to insert special offers

3.b.2 User Use Case Scenarios

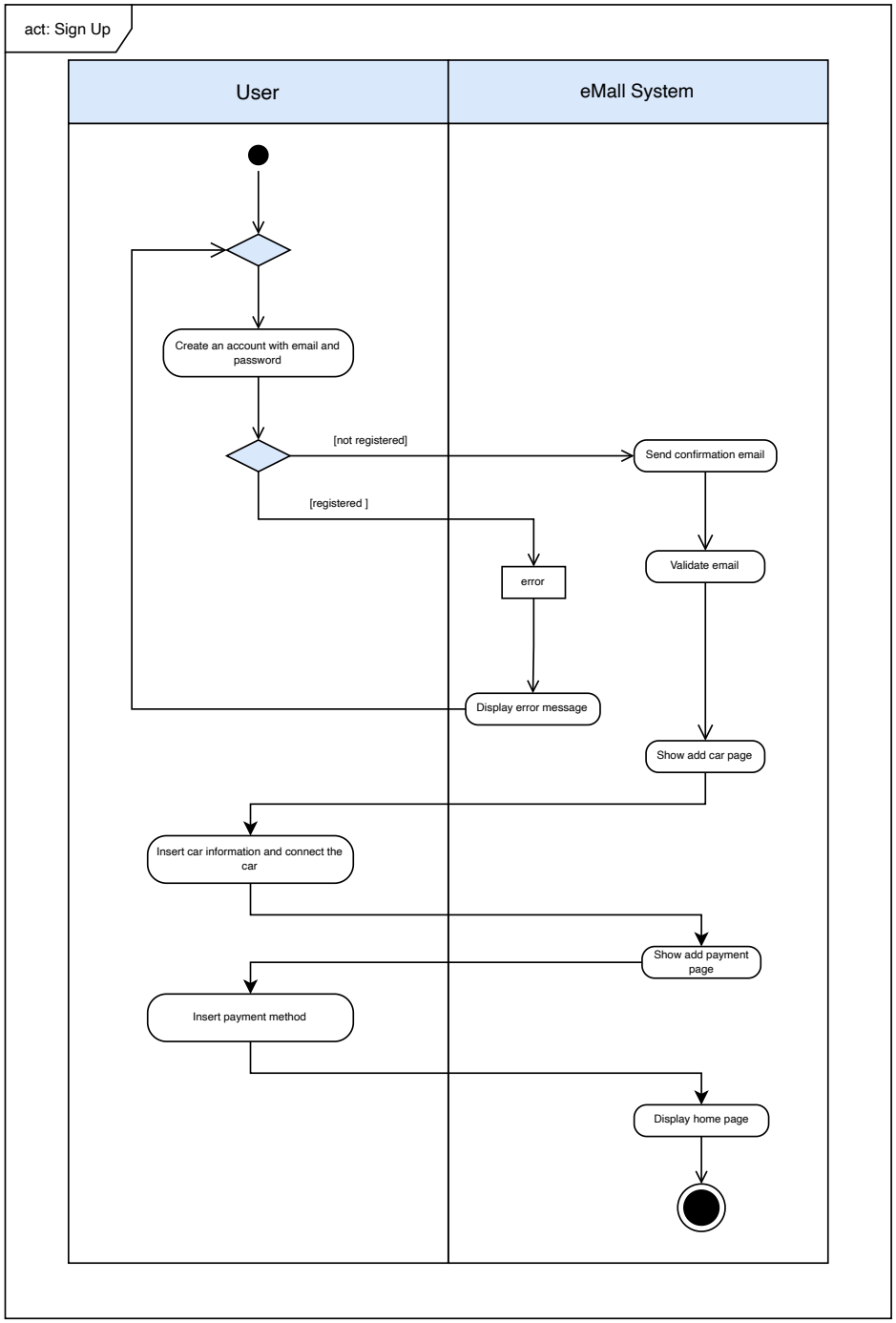
Unregistered user



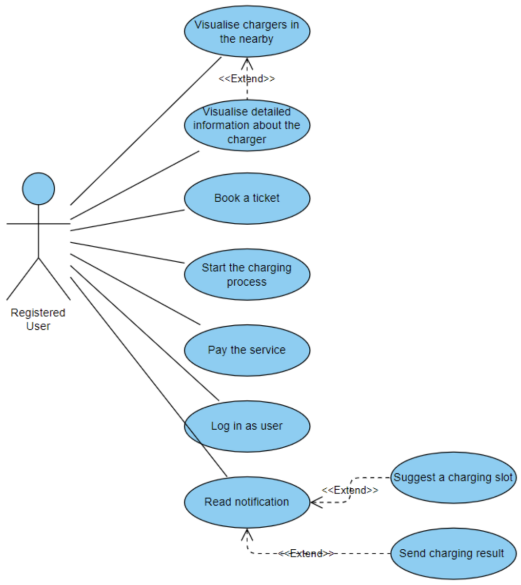
Sign Up

Name	Sign Up
ID	1
Actor	Customer
Entry conditions	• The application is running

Flow of events	1. The user presses "Sign up" button; 2. The user inserts the basic credentials (email, password); 3. The user inserts the car (or cars in case of multiple cars) data; 4. The user allows the GPS tracking by pressing the checkbox; 5. The user sends the data to the system by pressing the "Subscribe" button; 6. The user inserts his / her own credit card data 7. The system sends an email to the user on the mail inserted before;
Exit condition	The customer has clicked "confirm the mail" on the email received
Exception 1	User insert an email which is already stored in the eMall's database. So, after the user fills up the form and clicks on the confirmation button, the application displays an error message and invites the user, already registered, to login with that email.
Exception 2	User inserts an invalid email. So, after the user clicks on the confirmation button, the application displays an error message and invites the user to check his credentials or change them.


act: Sign Up
User
eMall System
Create an account with email and password
[not registered]
Send confirmation email
[registered]
Validate email
error
Display error message
Show add car page
Insert car information and connect the car
Show add payment page
Insert payment method
Display home page

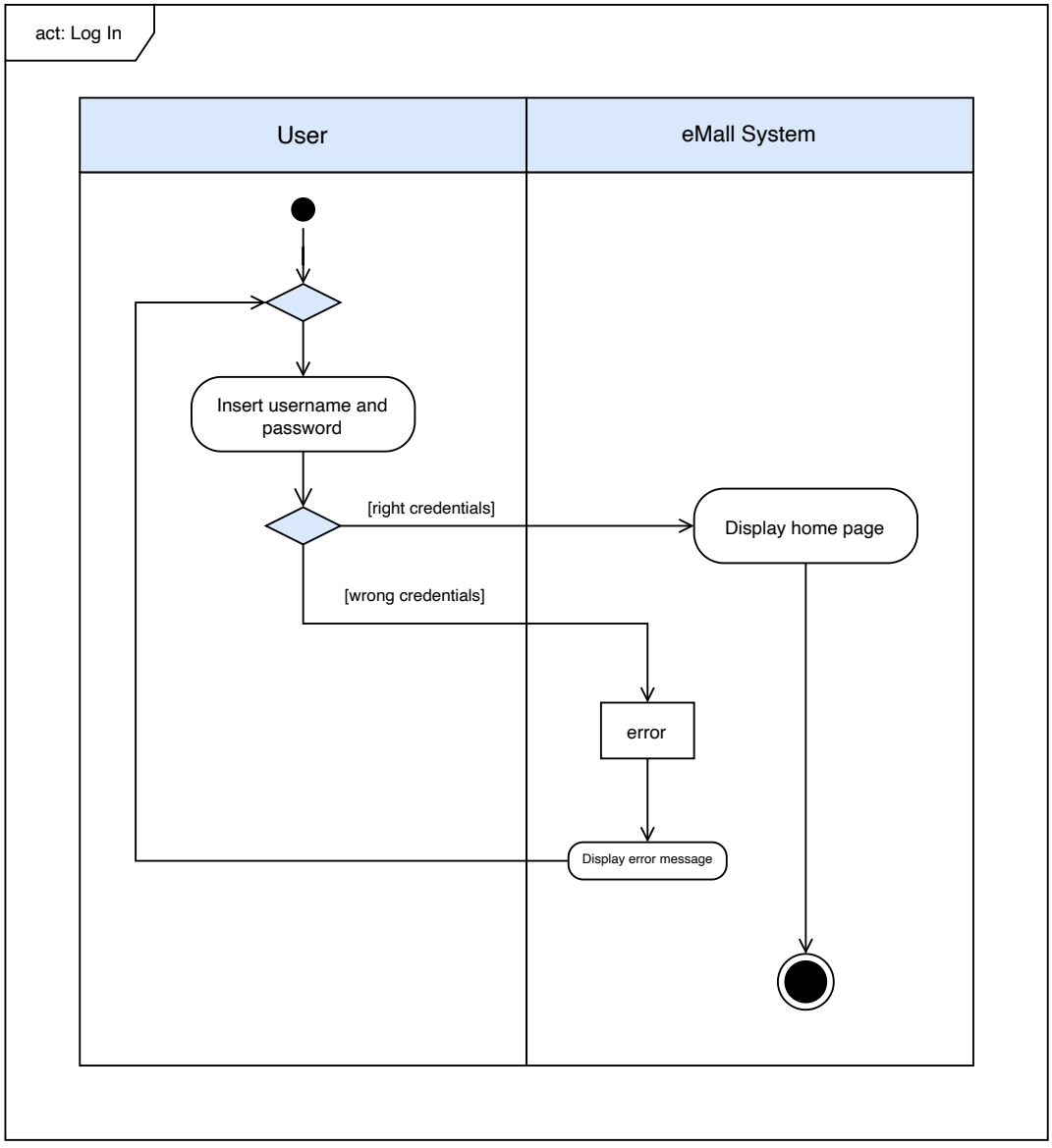
Registered user



Log In

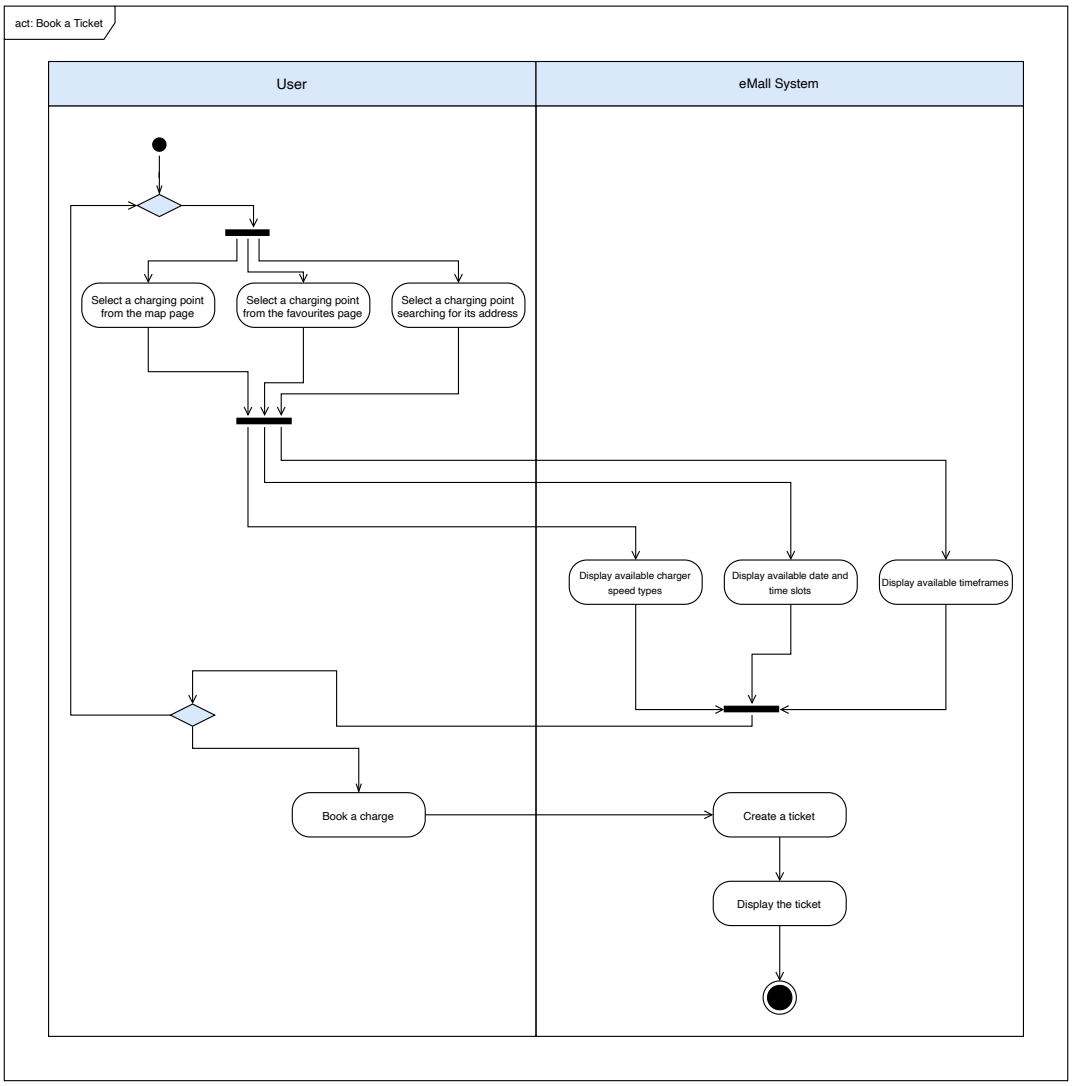
Name	Log in
ID	2
Actor	Customer
Entry conditions	• The application is running • The customer has been already registered
Flow of events	1. The user inserts the email 2. The user inserts the password
Exit condition	The user presses on the confirmation button and the map will be loaded

Exception	User inserts an invalid email and/or password. The application displays an error message and invites the user to change his credentials.



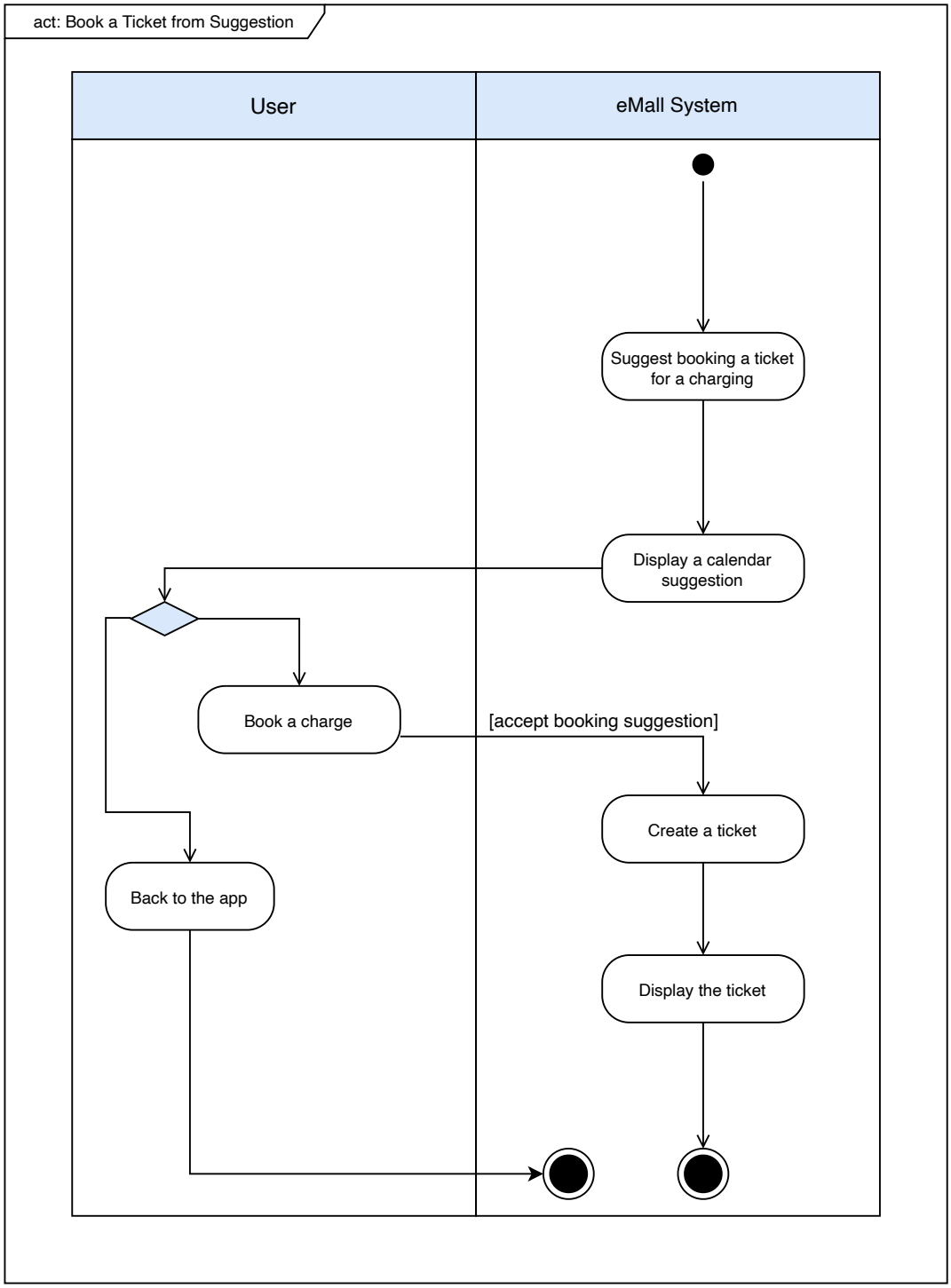
Booking a ticket

Name	Booking a ticket
ID	3
Actor	Customer
Entry conditions	• The application is running • The customer has been already logged • The GPS permission is granted
Flow of events	1. The user inserts the address in the bar, just loads its position or click on the favourites 2. The user clicks on the charging point 3. The user selects the speed of the charger with the yellow bar on the top 4. The user selects the date and the starting time on the calendar 5. The user selects the timeframe with the white bar below 6. The user presses on "Book a Ticket"
Exit condition	A ticket is created and it appears on the screen of the mobile phone



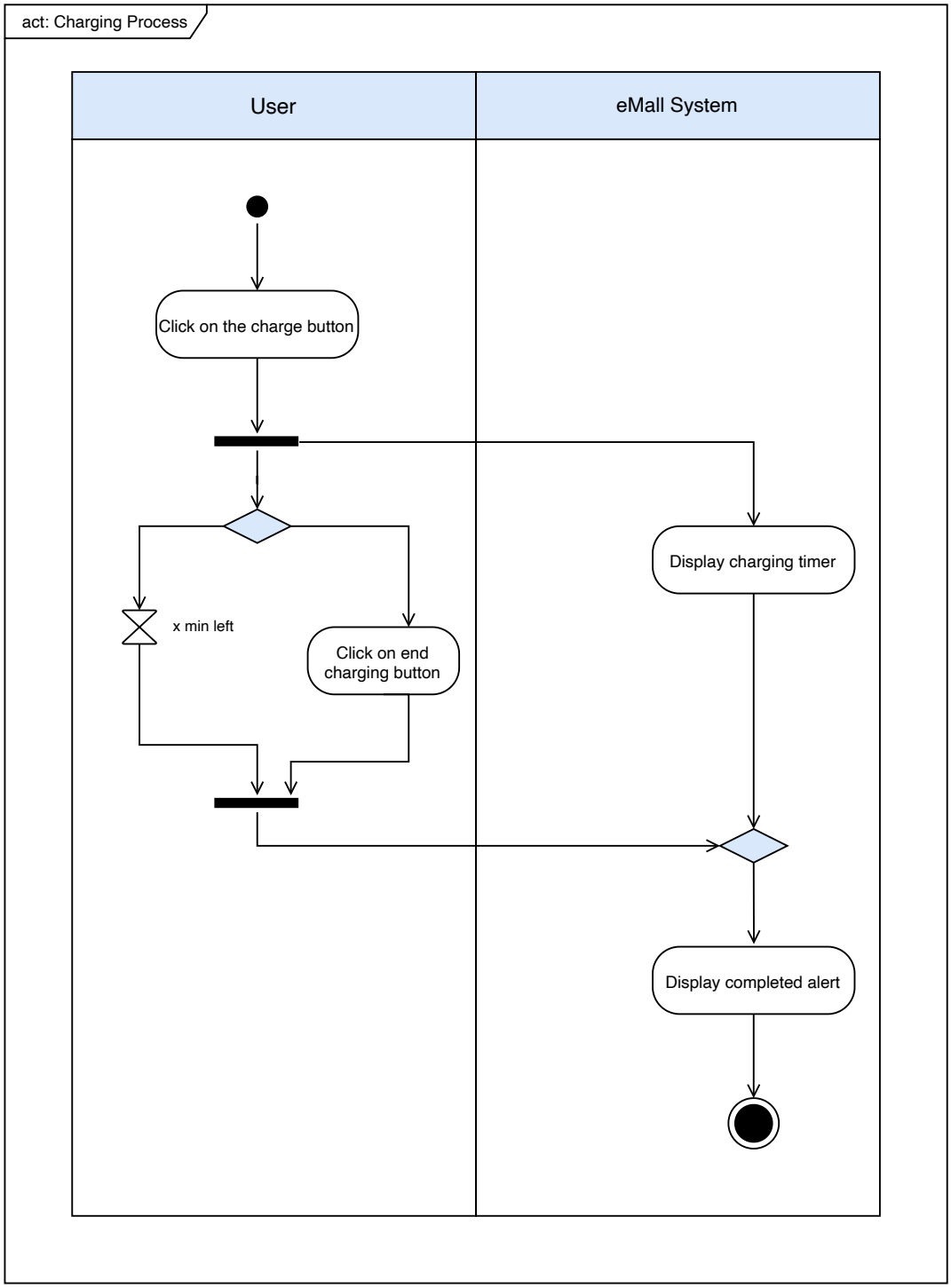
Name	Booking a ticket, calendar suggestion
ID	4
Actor	Customer

Entry conditions	• The application is running • The customer has been already logged
Flow of events	1. The user clicks on the notification “Calendar suggestion - e-mall” 2. The user presses on “Create a Ticket”
Exit condition	A ticket is created and it appears on the screen of the mobile phone


act: Book a Ticket from Suggestion
User
eMall System
Suggest booking a ticket for a charging
Display a calendar suggestion
Book a charge
[accept booking suggestion]
Create a ticket
Back to the app
Display the ticket

Charging process

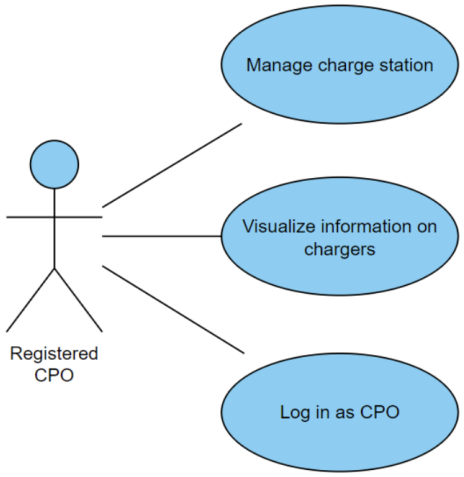
Name	Charging process
ID	5
Actor	Customer
Entry conditions	• The application is running • The customer has been already logged • The pump has been already inserted in the car
Flow of events	1. The user presses on "Tickets" button from the navbar below 2. The user presses on the "charge" button 3. The system will send a notification when the charging process will be finished
Exit condition	The user removes the pump from the car and the system withdraws the money from the user's credit card previously inserted
Exception	The user removes the pump before the end of the charging process. The system is notified of the action and it withdraws the money from the user's credit card


act: Charging Process
User
eMall System
Click on the charge button
x min left
Click on end charging button
Display charging timer
Display completed alert

3.b.3 CPO Use Case Scenarios

Unregistered CPO We suppose that all CPOs already have a working account and a new CPO account can be integrated only with an external API, due to the high-level of security needed by the software.

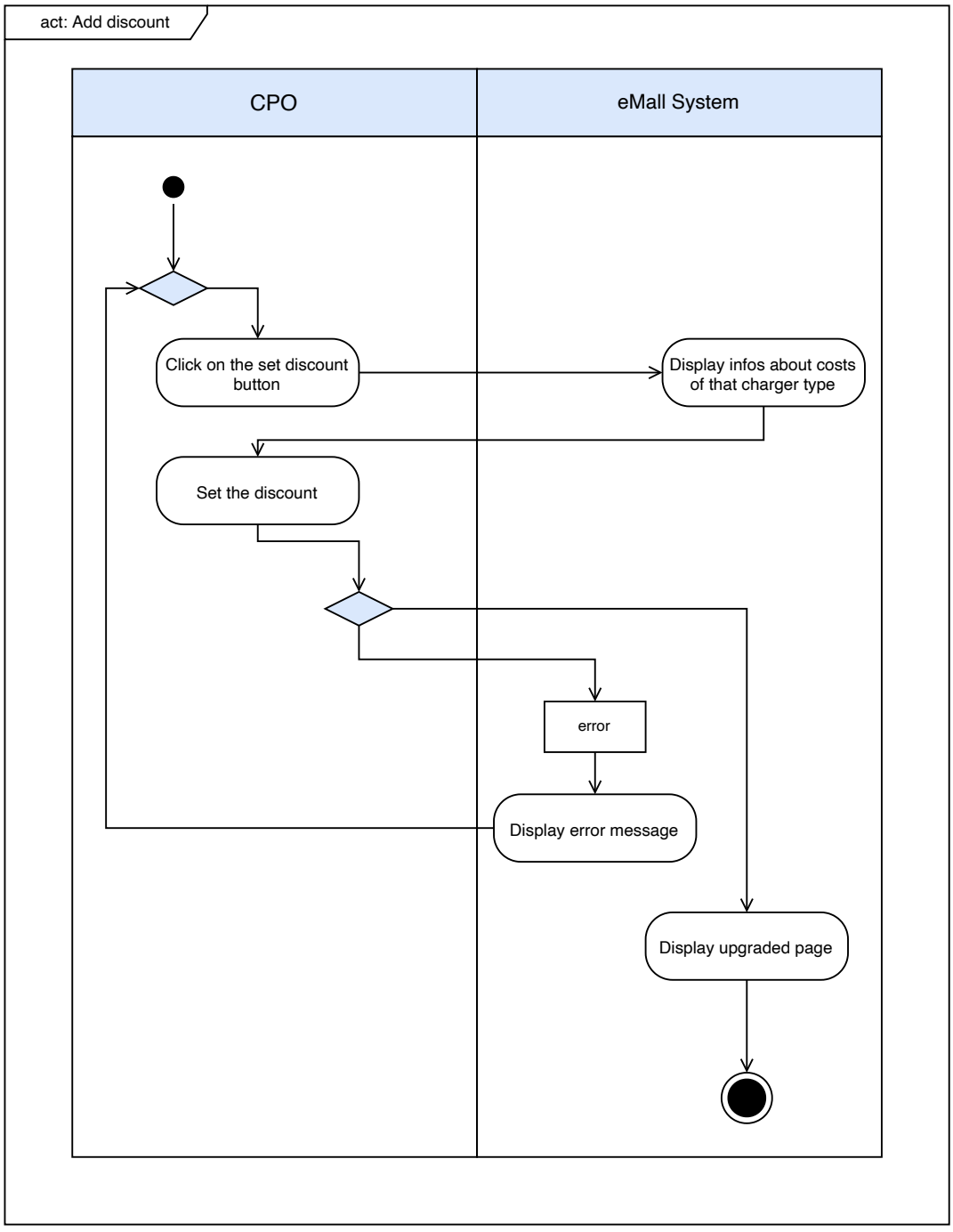
Registered CPO



Add discount

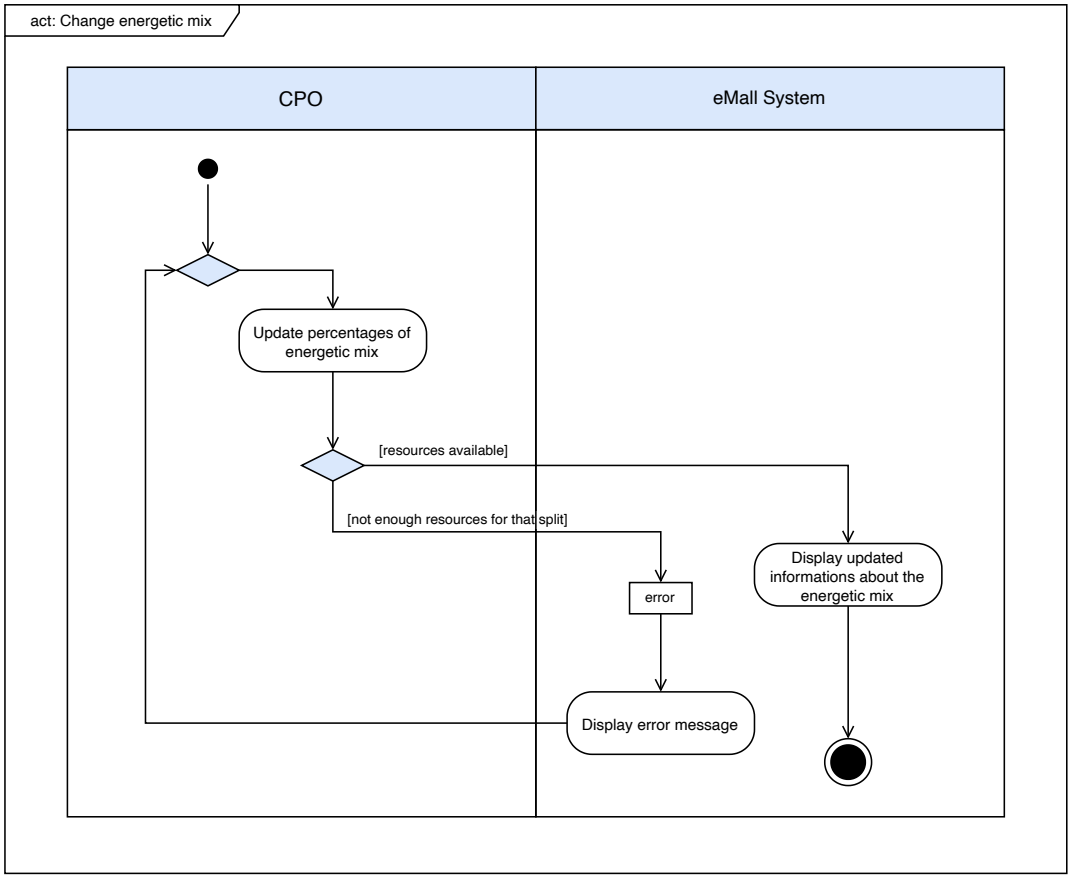
Name	Discount addiction
ID	6
Actor	CPO
Entry conditions	• The software is running • The CPO has been already logged

Flow of events	1. The CPO presses the "station" button 2. The CPO selects the charging station chosen 3. The CPO presses the "set discount" button on the right top of the charger overview 4. The CPO inserts the percentage of the discount 5. The CPO presses on the "create discount" button
Exit condition	The discount is notified to the charging point
Exception	CPMS can't reach the DBMS, so the discount can't be saved. So the CPO's page is reloaded with an error message display, asking him to retry to set the discount.


act: Add discount
CPO
eMall System
Click on the set discount button
Display infos about costs of that charger type
Set the discount
error
Display error message
Display upgraded page

Change energetic mix

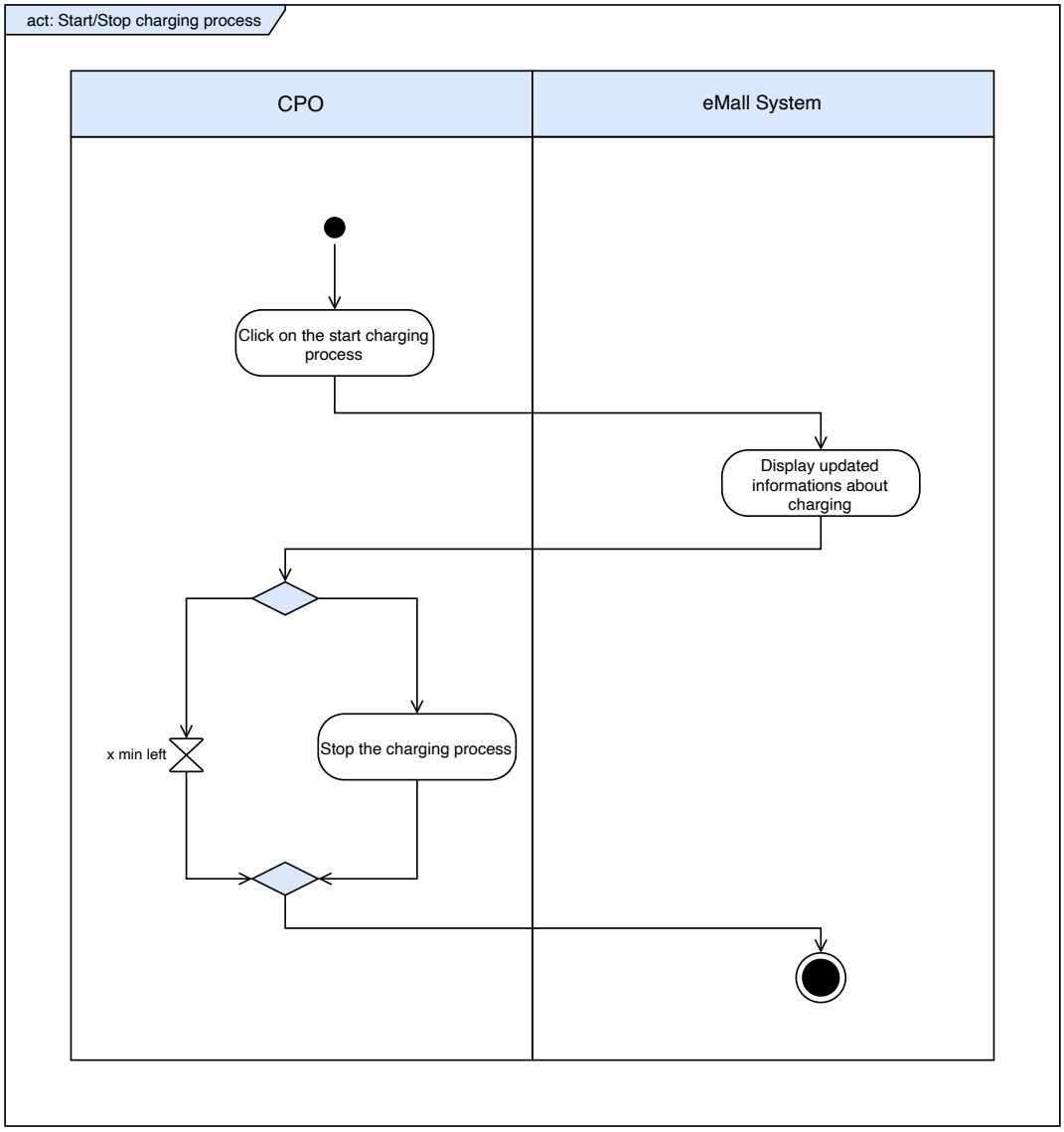
Name	Change energetic mix of a charging point
ID	7
Actor	CPO
Entry conditions	• The software is running • The CPO has been already logged
Flow of events	1. The CPO presses the "station" button 2. The CPO selects the charging station chosen 3. The CPO clicks on the bars next to "Renewable" and "no-Renewable" in the charging point type overview and changes the percentages 4. The CPO presses the "save" button, just appeared
Exit condition	The energetic mix change is notified to the charging point
Exception	The available resources of the station are not enough for the updated energetic mix split. The system shows an error message inviting the CPO to choose a split according to the actual status of the charging station or buy more resources.



Start/Stop the charging process

Name	Start/stop the charging process
ID	8
Actor	CPO
Entry conditions	• The software is running • The CPO has been already logged

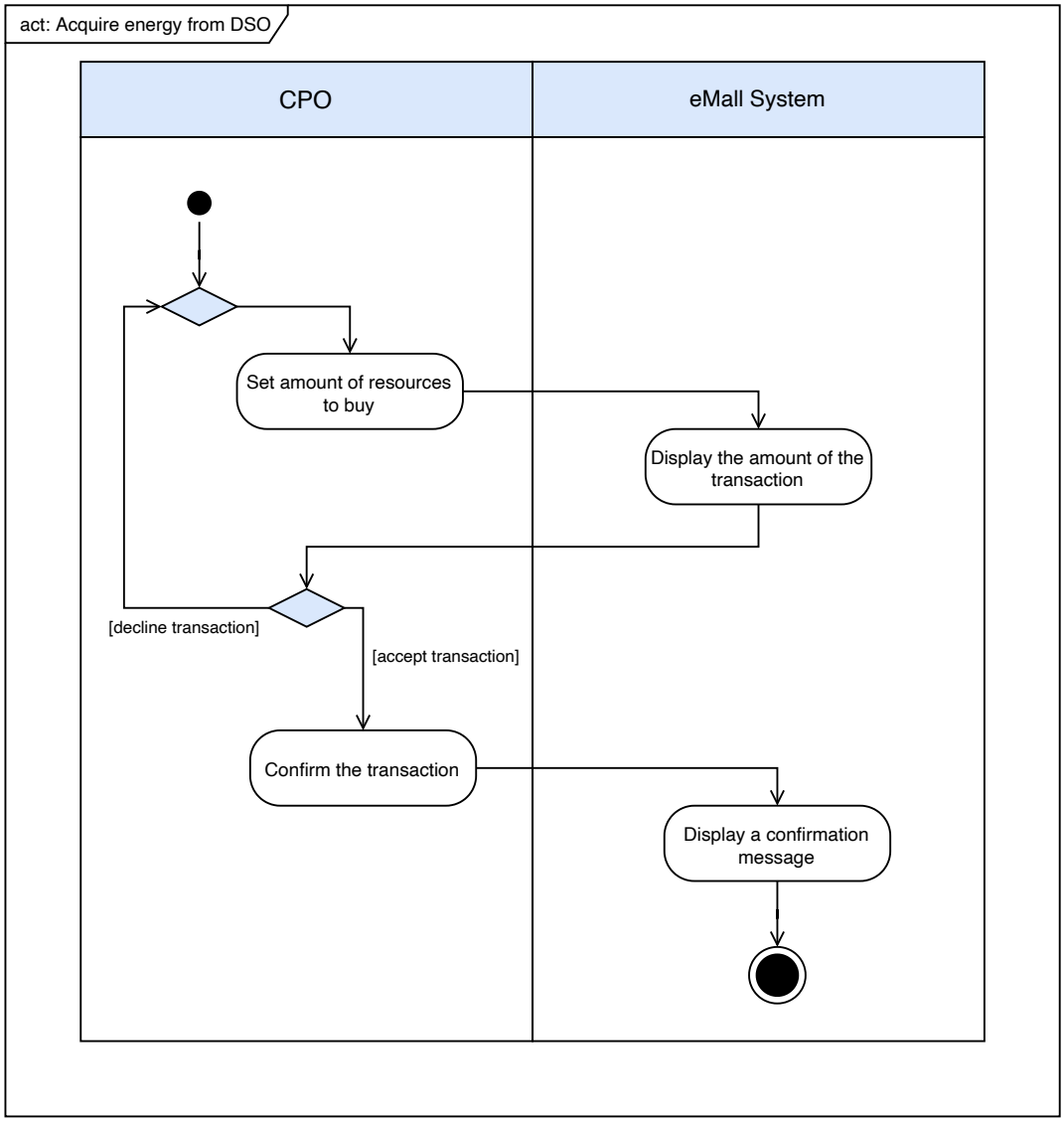
Flow of events	1. The CPO presses the “station” button 2. The CPO selects on the “start” or “stop” button to respectively start and stop the charging process
Exit condition	The charge activity change is notified to the charging point



Buy DSO energy

Name	Buy DSO energy
ID	9
Actor	CPO

Entry conditions	• The software is running • The CPO has been already logged
Flow of events	1. The CPO presses the "DSO Shop" button 2. The CPO selects the DSO from who wants to buy energy and select the percentage of energy 3. The CPO presses the "Choose" button to confirm the previous choice (in case he is changing DSO provider)
Exit condition	The DSO change is notified to charging stations and related providers



3.c Requirements Mapping

This subsection summarizes the goals described in section 1.1, showing the requirements and the domain assumptions for each of them. The main objectives of our system are:

- G1: Allow users to visualize chargers their cost and special offers.

  - R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
  - R2: The system allows the user to log in by entering the credentials used at the time of registration.
  - R3: The system must regularly update charging station data regarding cost and special offers
  - R4: The system allows the user to give permission about GPS location
  - R5: The system shall allow users to select a day, time and timeframe slot from the available ones
  - D4: The user must let the system to access the GPS
  - D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
  - D6: GPS position of the charging station is exact and leads to that charging station
  - D7: All users have access to internet connection

- G2: Allow users to book a reservation for a charge at a specific location and at a specific time and timeframe.
  - R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
  - R2: The system allows the user to log in by entering the credentials used at the time of registration.
  - R5: The system shall allow users to select a day, time and timeframe slot from the available ones
  - D1: The payment method must be correct
  - D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
  - D6: GPS position of the charging station is exact and leads to that charging station
  - D7: All users have access to internet connection

- G3: Allow users to start the charging process

  - G3.1: Allow users to start the charging process with the application
    * R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
    * R2: The system allows the user to log in by entering the credentials used at the time of registration.
    * R6: The system generates a TicketID associated to a charging reservation
    * R9: The system enables users to press the "start charging" button on the application
    * D1: The payment method must be correct
    * D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
    * D7: All users have access to internet connection
  - G3.2: Allow users to start the charging process with the screen of the charging station
    * R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
    * R2: The system allows the user to log in by entering the credentials used at the time of registration.
    * R6: The system generates a TicketID associated to a charging reservation
    * R8: The system enables users to insert the license plate onto a smart pump.
    * D1: The payment method must be correct
    * D5: Each user must insert correct data as input (i.e. car id, correct data, etc.

- G4: Let the system notify the user when the charging process is over
  - R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
  - R2: The system allows the user to log in by entering the credentials used at the time of registration.

  - R6: The system generates a TicketID associated to a charging reservation
  - D1: The payment method must be correct
  - D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
  - D7: All users have access to internet connection

- G5: Allow users to pay for the service
  - R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
  - R2: The system allows the user to log in by entering the credentials used at the time of registration.
  - R6: The system generates a TicketID associated to a charging reservation
  - R11: The system allows the user to change the payment method.
  - D1: The payment method must be correct
  - D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
  - D7: All users have access to internet connection

- G6: Let the system notify the user with calendar charging suggestions
  - R1: The system allows the user to sign up and provide their personal information, payment method and vehicle information.
  - R2: The system allows the user to log in by entering the credentials used at the time of registration.
  - R4: The system allows the user to give permission about GPS location
  - R7: The system computes a prediction of expected cost of the charge, based on the timeframe and charging info
  - R10: The system allows the user to connect its own calendar to the e-mall application

  - R11: The system recommends the user a ticket reservation in case it detects car's battery need by elaborating the best charging time to fit its calendar
  - D2: The events in the calendar must be correct
  - D3: The user must enter the locations of the events in his calendar
  - D5: Each user must insert correct data as input (i.e. car id, correct data, etc.)
  - D6: GPS position of the charging station is exact and leads to that charging station
  - D7: All users have access to internet connection
  - D8: The recommendations are based on the calendar of the user, especially on the time and position of his activities
  - D9: The recommendations are such that there are not interference in the same station

- G7: Allow the CPOs to globally manage their charge stations
  - R12: The system allows the CPO to know main information (occupied, type of charging, cost, time remaining, energy consumed, energy mix)
  - R13: The system allows the CPO to know main information related to the whole station (battery, status, number of cars in charge)
  - R14: The system allows the CPO to see the price of energy from the DSO
  - R15: The system allows the CPO to change the DSO provider and the energy mix for the whole station (all charging points)
  - R16: The system allows the CPO to insert special offers
  - D7: All users have access to internet connection
  - D10: The CPO must have an account

- G8: Allow the CPOs to manage charging points
  - R12: The system allows the CPO to know main information (occupied, type of charging, cost, time remaining, energy consumed, energy mix)

  - R13: The system allows the CPO to know main information related to the whole station (battery, status, number of cars in charge)
  - R14: The system allows the CPO to see the price of energy from the DSO
  - R15: The system allows the CPO to change the DSO provider and the energy mix for the whole station (all charging points)
  - R16: The system allows the CPO to insert special offers
  - D7: All users have access to internet connection
  - D10: The CPO must have an account
- G9: Let the system select on its own which DSO is the best to obtain energy from due to energy availability and cost
  - R12: The system allows the CPO to know main information (occupied, type of charging, cost, time remaining, energy consumed, energy mix)
  - R13: The system allows the CPO to know main information related to the whole station (battery, status, number of cars in charge)
  - R14: The system allows the CPO to see the price of energy from the DSO
  - D7: All users have access to internet connection

3.d Design Constraints

The System should be very intuitive and simple to use, as the range of users includes all demo-graphics and people with few digital knowledge.

3.d.1 Hardware Constraints

Every client needs a smartphone with the GPS location permission turned on. It allows you to reserve a charger for a specific period of time starting at a set time and locate nearby charging stations and related discounts.

Additionally, end users must manage how their car charges by the smartphone. The mobile application needs to alert users to the beginning and end.

As a result, on the side of the charging station, each pump must include hardware that not only enables the charging process when the car is connected to it, but (for each category) can send a request to a central server to dynamically select the best DSO(s) from which to draw energy.

3.d.2 Privacy Constraints

Location data is information that describes the precise location of a device, such as its latitude, longitude, or altitude, the user's direction of movement, or the time the location information was captured.

How to effectively use the data while respecting each user's need for privacy regarding their location is the main difficulty. Since the system's primary application area is in Europe, it must adhere to GDPR regulations.

Additionally, a secure connection must be used for data transmission to avoid Man in the Middle attacks.

An end user must read and approve the privacy statement before registering for the application. The user won't be able to access the service if this isn't the case.

4 Software System Attributes

Reliability Service fault tolerance is necessary to ensure continuity. To stop error propagation and data loss, methods for error management and fault containment must be set up.

The system must guarantee a 24/7 service.

Availability The system should be as available as possible, with a minimum value of 96% of time. To make this possible every updated version will be developed and available as a free update as soon as it's tested and ready. Also maintenance procedure will try to be the least intrusive possible with the system.

Security It will be stored users credentials and payment information. Security of the data and of the communications user-system and CPMS is a primary concern.

Maintainability The inevitable future changes to the system will be taken into account when designing the architecture. To assist and speed up any required update, several records of the design process and maintenance methods will be maintained.

Portability The application will be very responsive and adaptive in order to work on any mobile device running the iOS or Android operating system.

5 Formal Analysis using Alloy

In this section we will provide a formal model of the problem using Alloy. Then we will show some execution of the code to highlight different functionalities of the model, with their respective run worlds.

5.a Model

```
//------------------------------User
abstract sig AccountRegistration{
        emailAddress: one EmailAddress,
        password: one Password,
        name: one Name,
        surname: one Surname
}

sig EmailAddress{}
sig Name{}
sig Surname{}
sig Password{}
{//each password is associated to a registration but some registrations may
    ↪ have the same password
        all p : Password  | ( some r : AccountRegistration | r. password = p
            ↪ )
}

sig User extends AccountRegistration {
        car: some Car,
        paymentMethod: lone PaymentMethod,
        calendar: one Calendar,
        gpsPosition: one GpsPosition,
        notifications : set Notification,
        nearbyStations : set Station,
        favouritesStations: set Station,
        newTickets : set Ticket ,
        previousTickets : set Ticket
}

sig Car{
        licensePlate : one LicensePlate,
        autonomy: one Float,
        exactPosition : one GpsPosition,
        battery: one  Battery
}

sig LicensePlate{} //there can exists more Users with the same licensePlate
sig Battery{
        percentage : one Float,
        chargeStatus: one ChargeStatus
}

abstract sig ChargeStatus{}
one sig INCHARGE extends ChargeStatus{}
```

```
one sig DISCONNECTED extends ChargeStatus{}

sig Float {
        beforePoint : one Int ,
        afterPoint : one Int
}

sig  PaymentMethod{
        name : one Name,
        surname: one Surname,
        expireDate: one ExpireDate,
        CVC : one Int
}

sig ExpireDate{}
sig Calendar{
        freeTime : some Event,
        appointments : set Event
}

sig Event{
        startTime : one DateTime,
        duration : one Int,
        neighborhoodEventPosition : one GpsPosition
}

sig GpsPosition {}
{       // There can not exists more than one station with the same GPS
    ↪ position
        no disj s1 , s2 : Station | s2 . position = s1 . position
}

sig Ticket{
        station: one Station,
        booking: one Booking,
        ticketStatus : one TicketStatus,
        ticketID : one Int,
        chargeButton : one Button,
        timeLeft : one Float //to complete the recharge, 0 before the charge
            ↪  starts and 0 after it is completed
}
{
        ticketID > 0
        booking.reservedColumn in station.columns //the reservedColumn must
            ↪ be in the same station of the one named in the ticket
}

sig Booking{
        slot : one Slot,
        reservedColumn: one Column,
        approxCost: one Float
}

sig DateTime{}

sig Slot{
        reservation : one DateTime,
```

```
        timeframe : one Int
}

abstract sig Button{}
one sig STARTCHARGING extends Button{}
one sig STOPCHARGING extends Button{}

abstract sig TicketStatus{}
one sig ACTIVE extends TicketStatus{}
one sig EXPIRED extends TicketStatus{}
one sig CANCELLED extends TicketStatus{}

sig TicketsList{
        tickets : set Ticket
}
{
        no t : Ticket | t not in tickets
}

abstract sig Notification{}
{//a notification refers only to one user
        all n : Notification | (no disj u1 , u2 : User | n in u1 .
            ↪ notifications and n in u2 . notifications )
}

sig PAYMENT extends Notification{
        price : one Float,
        ticketId : one Int,
        dueTime: one DateTime
}
sig CHARGEFINISH extends Notification{
        endTime: one DateTime,
        timeToRescue: one Float
}

sig SUGGESTCHARGE extends Notification{
        suggestedTime: one DateTime,
        station: one Station
}

//------------------------------CPO
sig CPO extends AccountRegistration{
        workingStation: some Station,
        iban: one IBAN,
        idCard: one IdCard,
        phoneNumber: one PhoneNumber
}

sig IBAN{}
sig IdCard{}

sig PhoneNumber {}
{ // There can not exists more than one CPO with the same phone number
        no disj c1 , c2 : CPO | c2 . phoneNumber = c1 . phoneNumber
}
```

```
sig Station{
        columns : some Column,
        name: one Name,
        newTickets: set Ticket,
        batteries : set StationBattery,
        position : one GpsPosition,
        specialDeals : set SpecialDeals,
        choosenDSO : one DSO,
        stationStatus : one StationStatus,
        dsoList: some DSO
}
{
        no b : StationBattery | b not in batteries
        no d : DSO | d not in dsoList
}

abstract sig StationStatus{}
one sig FREE extends StationStatus{} //at least 1 free column
one sig FULL extends StationStatus{} //all columns are occupied

sig StationBattery {
        autonomy : one Float
}

sig SpecialDeals{
        discountedColumns : some Column,
        deal : one Float,
        duration : one Int,
        dealState : one DealState
}

abstract sig DealState{}
one sig ACTIVEDEAL extends DealState{}
one sig NOTACTIVEDEAL extends DealState{}

sig Column{
        connectedCar : lone Car,
        consumedEnergy: one Float,
        timeLeft : one Float, //time left to the column to become free
        columnStatus : one ColumnStatus,
        forceButton : one Button,
        energyMix : one Energy,
        type: one ColumnType,
        price : one Float,
        chargerNumber: one Int,
        renewablePercentage : one Int,
}
{
        price.beforePoint >0
}

abstract sig ColumnStatus{}
one sig FREECOLUMN extends ColumnStatus{}
one sig OCCUPIEDCOLUMN extends ColumnStatus{}
one sig EMITTINGCOLUMN extends ColumnStatus{}

abstract sig Energy{}
```

```
one sig FROMDSO extends Energy{} //take energy from DSO
one sig FROMBATTERY extends Energy{} //take energy from station batteries
sig FROMMIX extends Energy{//take energy from a Mix of batteries and DSO
        dsoPercentage : one Int,
        batteryPercentage : one Int
}

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

sig DSO{
        name: one Name,
        energyPrice : one Float,
        energyType : one EnergyType,
        avaiability : one Avaiability
}

abstract sig EnergyType{} //It is the predominant energy type adopted by the
    ↪  DSO
one sig COAL extends EnergyType{}
one sig GAS extends EnergyType{}
one sig RENEWABLE extends EnergyType{}
one sig NUCLEAR extends EnergyType{}
one sig OIL extends EnergyType{}

abstract sig Avaiability{}
one sig AVAIABLE extends Avaiability{}
one sig NOTAVAIABLE extends Avaiability{}


//
    ↪ ------------------------------------------------------------------------------------------
    ↪
//Facts
fact { //a unique email address is associated to each registration
        no disj r1 , r2 : AccountRegistration | r1 . emailAddress = r2 .
            ↪ emailAddress
}

fact { // for each user , the set of bookings must be different
        all disj u1 , u2 : User | ( u1 . previousTickets + u1 . newTickets )
            ↪  & ( u2 . previousTickets + u2 . newTickets ) = none
}

fact { //a user can not have two bookings at the same time
        all u : User | (no disj t1 , t2 : Ticket | t1 in u. newTickets and
            ↪ t2 in u. newTickets and t1 . booking . slot . reservation =
            ↪ t2 . booking . slot .reservation )
}

fact { //a ticket refers only to one user
        all t : Ticket | (no disj u1 , u2 : User | (t in u1 . newTickets and
            ↪  t in u2 . newTickets ) or (t in u1 . previousTickets and t
            ↪ in u2. previousTickets ))
}
```

```
fact { //a ticket always has a user
        all t : Ticket | ( one u : User | t in u. newTickets or t in u .
            ↪ previousTickets )
}

fact { // each user has disjoint upcoming and past bookings
        all u : User | u. newTickets & u. previousTickets = none
}

fact { // each ticket id is unique
        no disj t1 , t2 : Ticket | t1 . ticketID = t2 .ticketID
}

fact { //A booking refers only to one ticket
        no disj t1 , t2 : Ticket | t1 . booking = t2 . booking
}

fact { //A booking cannot exist without an associated ticket
        all b : Booking | one t : Ticket | t. booking = b
}

fact { // There can not exist one slot without any booking
        all s : Slot | some b : Booking | b. slot = s
}

fact { // Each station has at most one CPO
        no disj cpo1 , cpo2 : CPO | cpo1.workingStation = cpo2 .
            ↪ workingStation
}

fact { // Each user has at most one payment method
        no disj u1 , u2 : User | u1.paymentMethod= u2.paymentMethod
}

fact { // Each user has at most one calendar
        no disj u1 , u2 : User | u1.calendar= u2.calendar
}

fact { //A cancelled or used ticket cannot be in upcoming tickets
        all u : User | (all t : Ticket | t.ticketStatus ≠ ACTIVE implies u.
            ↪ newTickets & t = none )
}

fact { // An active ticket cannot be in past tickets
        all u : User | (no t : Ticket | t. ticketStatus = ACTIVE and t  in (
            ↪ u. previousTickets) )
}

fact { // If a ticket exists , it must be active, expired or cancelled
        all t : Ticket | t. ticketStatus = ACTIVE or t. ticketStatus =
            ↪ EXPIRED or t.ticketStatus = CANCELLED
}

fact { // The station name must be unique
        no disj s1 , s2 : Station | s1 . name = s2 . name
}
```

```
fact { // If a ticket is active , it must have a duration greater then 0
        all t : Ticket |  not(t. ticketStatus = ACTIVE and t . booking .
            ↪ slot .timeframe = 0)
}

fact { //A station can not exist without an CPO
        all s : Station | one cpo : CPO | cpo.workingStation = s
}

fact { //A DateTime is always associated to a slot
        all s : Slot | one d : DateTime | s. reservation = d
}

fact{//A car with a battery of 100% can't be INCHARGE
        all c : Car | not ( c.battery.chargeStatus = INCHARGE and c.battery.
            ↪ percentage.beforePoint = 100)
}

fact{//There is no car without an owner
        no c : Car | ( all u : User | c not in u.car)
}

fact { //A ticket can be a new ticket only if it is active
        all u : User | all t : Ticket | not(t in u.newTickets) and t.
            ↪ ticketStatus = EXPIRED or t.ticketStatus = CANCELLED
}

fact {//Two cars cannot have the same battery
        no disj c1, c2 : Car | c1.battery = c2.battery
}

fact { //start chariging when it is needed
        all u : User | all c : Column | all v : Car | ((c in u.newTickets.
            ↪ booking.reservedColumn) and (v in u.car) and u.newTickets.
            ↪ chargeButton = STARTCHARGING and (u.newTickets.station.
            ↪ position = v.exactPosition and u.newTickets.timeLeft.
            ↪ beforePoint > 0)) implies (v.battery.chargeStatus = INCHARGE
            ↪ and c.columnStatus = EMITTINGCOLUMN)
}

fact{ //if a column is emitting energy it must consume energy
        all c : Column | c.columnStatus = EMITTINGCOLUMN and c.
            ↪ consumedEnergy.beforePoint > 0
}

fact{ //The approximate cost of a ticket can not be 0
        all t : Ticket | not (t.booking.approxCost.beforePoint = 0)
}

fact { //a unique IBAN is associtaed to each CPO
        no disj r1 , r2 : CPO | r1. iban = r2 .iban
}

fact { //a unique idCard is associated to each CPO
        no disj r1 , r2 : CPO | r1. idCard = r2 .idCard
}
```

```
fact{ //a unique chargerNumber is associated to each Column
        no disj c1,c2 : Column | c1.chargerNumber = c2.chargerNumber
}

fact { //a column refers only to one station
        all c : Column | (no disj s1 , s2 : Station | (c  in s1 .columns and
            ↪  c in s2 . columns ) )
}

fact{ //the energy mix of a column that use energy from mix sources can not
    ↪ be full renewable if the energyType is not renewable
        all s : Station | not (s.columns.renewablePercentage = 100 and s.
            ↪ columns.energyMix = FROMMIX and
        (s.choosenDSO.energyType = COAL or s.choosenDSO.energyType = OIL or
            ↪ s.choosenDSO.energyType = GAS))
}

fact{ //the energy mix of a column that use energy from dso has 0 renewable
    ↪ percentage if the source it is not renwable
        all s : Station | s.columns.renewablePercentage = 0 and s.columns.
            ↪ energyMix = FROMDSO  and
        (s.choosenDSO.energyType = COAL or s.choosenDSO.energyType = OIL or
            ↪ s.choosenDSO.energyType = GAS)
}

fact{ //the energy mix of a column that use energy from dso has 100
    ↪ renewable percentage if the source it is renwable
        all s : Station | (s.choosenDSO.energyType = RENEWABLE or s.
            ↪ choosenDSO.energyType = NUCLEAR and s.columns.energyMix =
            ↪ FROMDSO) implies (s.columns.renewablePercentage = 100)

}

fact{//A column is occupied when finish its charge, the car is disconnected
    ↪ but the car is still in there
        all u : User | all s : Station | all c : Column | (c in u.newTickets
            ↪ .booking.reservedColumn and s in u.newTickets.station and  u.
            ↪ newTickets.timeLeft.beforePoint = 0 and (u.car.exactPosition
            ↪ = s.position ))  and u.car.battery.chargeStatus =
            ↪ DISCONNECTED implies c.columnStatus = OCCUPIEDCOLUMN
}

fact { //A ticket can be a previous ticket only if it is EXPIRED or
    ↪ CANCELLED
        all u : User | all t : Ticket | (t in u.previousTickets) implies (t.
            ↪ ticketStatus = EXPIRED or t.ticketStatus = CANCELLED)
}

fact{//A column is free when finish its charge and the car frees the column
        all u : User | all s : Station | all c : Column | (c in u.newTickets
            ↪ .booking.reservedColumn and s in u.newTickets.station and  u.
            ↪ newTickets.timeLeft.beforePoint = 0 and not(u.car.
            ↪ exactPosition = s.position )) implies c.columnStatus =
            ↪ FREECOLUMN
}
```

```
fact{//a unique ticketId is associated to each PAYMENT
        no disj p1, p2 : PAYMENT | p1.ticketId = p2.ticketId
}

fact{//each payment is associated to the related ticketID
        all p : PAYMENT |  one t : Ticket | t.ticketID = p.ticketId
}

fact { //a specialDeal is always associated with a Station
        all sD : SpecialDeals | (one s : Station |  sD in s.specialDeals)
}

fact{ //there can not be a car with autonomy > 0 and battery percentage = 0
        all c : Car | not(c.autonomy.beforePoint > 0 and c.autonomy.
            ↪ afterPoint > 0 and c.battery.percentage.beforePoint = 0)
}

fact { //a user can not have two appointments at the same time
        all u : User | (no disj e1 , e2 : Event | e1 in u. calendar.
            ↪ appointments and e2 in u. calendar. appointments and e1 .
            ↪ startTime = e2 . startTime )
}

fact { //a user can not have appointments and freeTime at the same time
        all u : User | (no disj e1 , e2 : Event | e1 in u. calendar.
            ↪ freeTime and e2 in u. calendar. appointments and e1 .
            ↪ startTime = e2 . startTime )
}

fact{//a charge is suggested when there is a special deal or the car
    ↪ autonomy is < 50 and the user has freeTime or has an appointment in
    ↪ the same gspPosition (neighborhood) of the chargeStation and it has
    ↪ free columns
        all s : SUGGESTCHARGE | one u : User | (u. car. autonomy .
            ↪ beforePoint < 50 or u.favouritesStations.specialDeals.
            ↪ dealState = ACTIVEDEAL) and ((s.suggestedTime = u.calendar.
            ↪ freeTime.startTime or (s.suggestedTime = u.calendar.
            ↪ appointments.startTime and u.calendar.appointments.
            ↪ neighborhoodEventPosition = s.station.position)) and s.
            ↪ station.stationStatus = FREE)
}

fact{ //all the notifications are related to a user
        all n : Notification | ( one u : User | n in u. notifications)
}

fact{//there is no column which is occupied and its status is free
                no c : Column | #(c.connectedCar)=1 and c.columnStatus =
                    ↪ FREECOLUMN
}

fact{//there is no column which is free and its status is occupied
                no c : Column | #(c.connectedCar)=0 and c.columnStatus =
                    ↪ OCCUPIEDCOLUMN
}

fact{ //if there is at least 1 free column stationStaus is free
```

```
        all s : Station | some c : Column | c in s.columns and c.
            ↪ columnStatus = FREECOLUMN implies (s.stationStatus = FREE)
}

fact{ //if there isn't at least 1 free column stationStaus is full
        all s : Station | all c : Column | c in s.columns and c.columnStatus
            ↪  = OCCUPIEDCOLUMN implies (s.stationStatus = FULL)
}

fact{ //by default the choosen DSO of a station is the one with the lower
    ↪ price, best enery type and which is avaible
        all disj d1, d2 : DSO | all s : Station | s.choosenDSO = d1 and ((d1
            ↪ .energyPrice.beforePoint < d2.energyPrice.beforePoint) and (
            ↪ d1.energyType = RENEWABLE or d1.energyType = NUCLEAR) and d1.
            ↪ avaiability = AVAIABLE)
}


fact{ //allow the CPO to force the charging start
        all c : CPO | all col : Column | col in c.workingStation.columns and
            ↪  (c.workingStation.columns.forceButton = STARTCHARGING) and
            ↪ col.columnStatus = EMITTINGCOLUMN
}

fact{//allow the CPO to force charging stop
        all c : CPO | all col : Column | col in c.workingStation.columns and
            ↪  (c.workingStation.columns.forceButton = STOPCHARGING)
            ↪ implies col.columnStatus = OCCUPIEDCOLUMN
}

fact{ //if a deal is active it must have a duration
        no s : SpecialDeals | s.dealState = ACTIVEDEAL and s.duration = 0
}


fact{//if a deal is not active it must not have a duration
        no s : SpecialDeals | s.dealState = NOTACTIVEDEAL and s.duration > 0
}

//
    ↪ ----------------------------------------------------------------------------------------------------
    ↪
//Assertions

//G2 : Allow users to book a reservation for a charge at a specific location
    ↪  and at a specific time and timeframe.
assert createATicket {
        all t : Ticket | t. ticketStatus = ACTIVE implies ( one u : User | t
            ↪  in u. newTickets and t . booking.slot .timeframe > 0)
}
check createATicket for 4

//G5: Allow users to pay for the service
assert paymentNotification{
        all p : Notification | p = PAYMENT implies (one u : User | p.
            ↪ ticketId in u.previousTickets.ticketID)
}
```

```
check paymentNotification for 4

//G6 : Let the system notify the user with calendar charging suggestions
assert suggestNotification{
        all s : Notification | s = SUGGESTCHARGE implies (one u : User | (u.
            ↪  car. autonomy . beforePoint < 50 or u.favouritesStations.
            ↪ specialDeals.dealState = ACTIVEDEAL) and s.suggestedTime = u.
            ↪ calendar.freeTime.startTime)
}
check suggestNotification for 1


//G7: Allow the CPOs to globally manage their charge stations
//In particular, allow the CPO to add a special deal
assert addSpecialDeal{
        no sD : SpecialDeals | one s : Station | sD not in s.specialDeals
}
check addSpecialDeal for 4

//G9: Let the system select on its own which DSO is the best to obtain
    ↪ energy from due to energy availability and cost
assert chooseDSO{
        all s : Station | all disj d1, d2 : DSO | s.choosenDSO in d1 implies
            ↪  (d1.energyPrice.beforePoint < d2.energyPrice.beforePoint)
            ↪ and (d1.energyType = RENEWABLE or d1.energyType = NUCLEAR)
            ↪ and d1.avaiability = AVAIABLE
}
check chooseDSO for 4

pred world1{
        #User = 2
        #Car = 3
        #Station = 1
}
run world1 for 4

pred world2{
        #User = 1
        #Car = 1
        #Station = 1
        #Column = 3
        #Ticket = 2
        #SpecialDeals = 0
}
run world2 for 4

pred world3{
        #User = 2
        #Car = 1
        #Station = 1
        #Notification = 3

        #Battery = 1
}
run world3 for 4

pred world4{
```

```
        #DSO = 1
        #CPO≥ 1
        #Station ≥ 1
        #Column> 1
        #StationBattery≥ 1
        #Calendar = 0
}
run world4 for 4
```

5.b Worlds

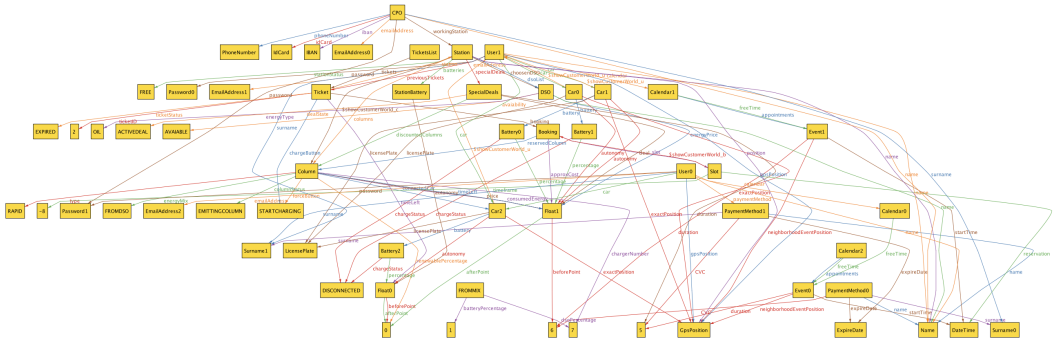
Below there is the execution result of the model and the worlds.

```
Executing "Check createATicket for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  33409 vars. 1962 primary vars. 73815 clauses. 357ms.
  No counterexample found. Assertion may be valid. 9ms.

Executing "Check paymentNotification for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  66854 vars. 3924 primary vars. 147863 clauses. 282ms.
  No counterexample found. Assertion may be valid. 124ms.

Executing "Check suggestNotification for 1"
  Solver=sat4j Bitwidth=4 MaxSeq=1 SkolemDepth=1 Symmetry=20 Mode=batch
  69852 vars. 4207 primary vars. 154682 clauses. 84ms.
  No counterexample found. Assertion may be valid. 0ms.

Executing "Check addSpecialDeal for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  102981 vars. 6169 primary vars. 227493 clauses. 284ms.
  No counterexample found. Assertion may be valid. 20ms.

Executing "Check chooseDSO for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  136701 vars. 8139 primary vars. 302254 clauses. 287ms.
  No counterexample found. Assertion may be valid. 8ms.

Executing "Run world1 for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  169804 vars. 10097 primary vars. 375044 clauses. 285ms.
  Instance found. Predicate is consistent. 52ms.

Executing "Run world2 for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  202943 vars. 12055 primary vars. 447957 clauses. 296ms.
  Instance found. Predicate is consistent. 48ms.

Executing "Run world3 for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  236070 vars. 14013 primary vars. 520829 clauses. 292ms.
  Instance found. Predicate is consistent. 57ms.

Executing "Run world4 for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  269219 vars. 15971 primary vars. 593748 clauses. 320ms.
  Instance found. Predicate is consistent. 38ms.
```

```
9 commands were executed. The results are:
  #1: No counterexample found. createATicket may be valid.
  #2: No counterexample found. paymentNotification may be valid.
  #3: No counterexample found. suggestNotification may be valid.
  #4: No counterexample found. addSpecialDeal may be valid.
  #5: No counterexample found. chooseDSO may be valid.
  #6: Instance found. world1 is consistent.
  #7: Instance found. world2 is consistent.
  #8: Instance found. world3 is consistent.
  #9: Instance found. world4 is consistent.
```

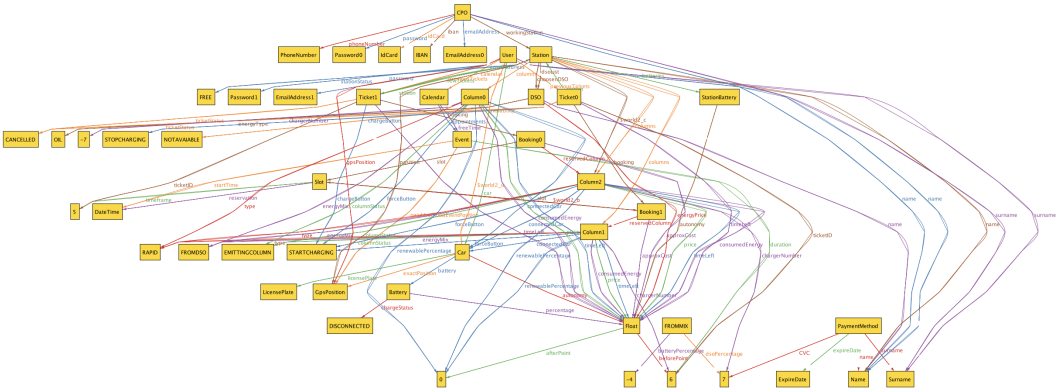
World 1 The first world focuses on the creation of the main actors. It is composed of users, with cars and a CPO with his station. This world permits to show the creation of the main objects of the model.

```
pred world1{
        #User = 2
        #Car = 3
        #Station = 1
}
run world1 for 4
```



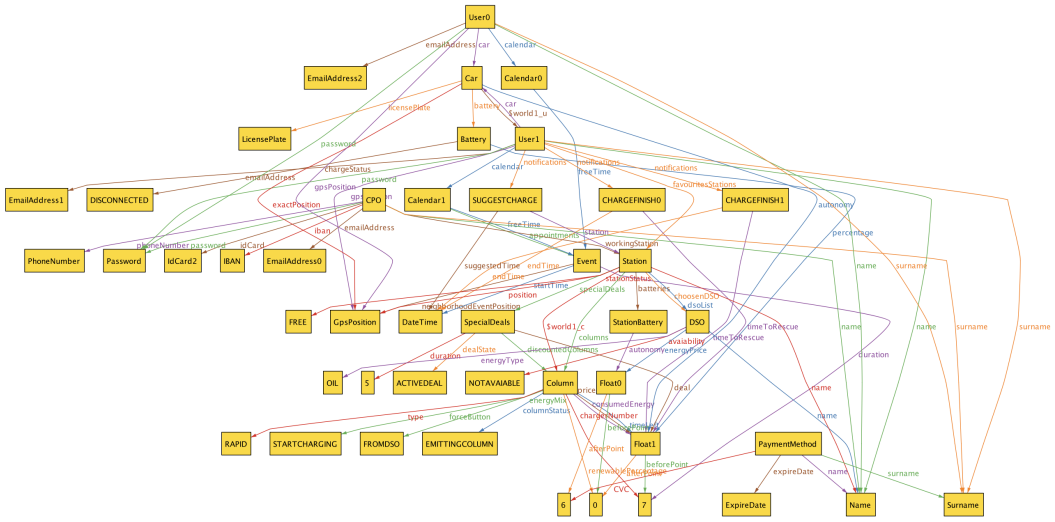
World 2 The second world focuses on the booking ticket process. It is composed of a user with some related tickets and a CPO with his station. This worlds permits to show the ticketing system between users and stations.

```
pred world2{
        #User = 1
        #Car = 1
        #Station = 1
        #Column = 3
        #Ticket = 2
        #SpecialDeals = 0
}
run world2 for 4
```



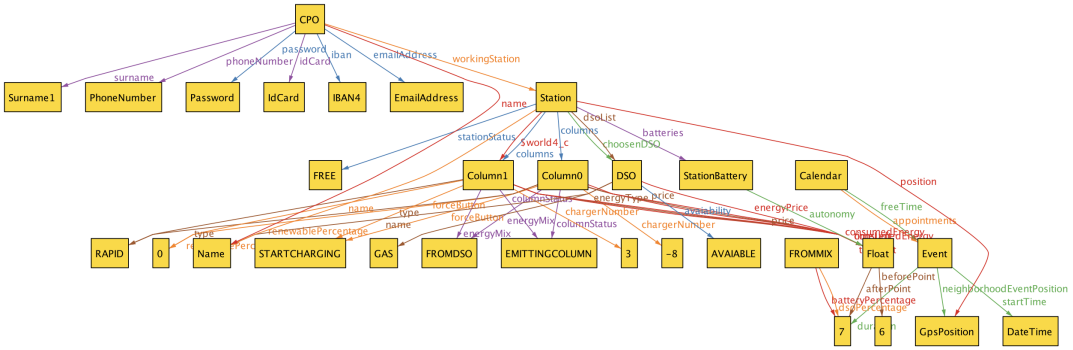
World 3 The third world focuses on the suggestion process. In particular, in this world, the system suggests a booking to a user, whose car needs a charge, and this booking is also related to a special deal of a charging station.

```
pred world3{
        #User = 2
        #Car = 1
        #Station = 1
        #Notification = 3

        #Battery = 1
}
run world3 for 4
```



World 4 The fourth world focuses on the CPO. It is composed of a CPO, his station, with all the detail related and an available DSO, from which he can buy energy.

```
pred world4{
        #DSO = 1
        #CPO≥ 1
        #Station ≥ 1
        #Column> 1
        #StationBattery≥ 1
}
run world4 for 4
```


CPO
Surname1
PhoneNumber
Password
IdCard
IBAN4
EmailAddress
Station
surname
phoneNumber
password
idCard
iban
emailAddress
workingStation
name
stationStatus
world4
c
columns
dsoList
batteries
choosenDSO
FREE
Column1
Column0
DSO
StationBattery
Calendar
position
RAPID
0
Name
STARTCHARGING
GAS
FROMDSO
EMITTINGCOLUMN
3
-8
AVAIABLE
FROMMIX
Float
Event
type
forceButton
energyMix
columnStatus
chargerNumber
energyType
price
availability
energyPrice
autonomy
freeTime
appointments
consumedEnergy
renewablePercentage
beforePoint
afterPoint
batteryPercentage
7
6
GpsPosition
DateTime
neighborhoodEventPosition
startTime

6 Effort Spent

Student	S.1	S.2	S.3	S.4
Valeria Amato	2h	7h	13h	10h
Francesco Dettori	2h	10h	5h	15h
Matteo Pancini	2h	5h	14h	5h