Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Computer Science and Engineering

Software Engineering 2 Project

eMall - eMSP system

Requirement Analysis and Specification Document

Authors:
Federico Bono
Daniele Cipollone

Academic Year 2022/2023

Milano, 21/12/2022
Version 1.1

Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to analyze and define the goals and requirements to later design, on behalf of the e-Mobility Service Providers (eMSPs), the infrastructure for the eMall App.

1.1.1 Goals

G1	Allow users to find charging stations nearby and their tariffs
G2	Allow users to book a charge session in one of the Charging Point
G3	Allow users to authenticate to the Charge Point and start the charging session
G4	Allow users to check the status of an active charging session
G5	Allow users to be notified when the charging process is completed
G6	Allow users to pay for the charging session

1.2 Scope

Electric mobility (e-Mobility) is a way to limit the carbon footprint caused by our urban and sub-urban mobility needs. When using an electric vehicle, knowing where to charge the vehicle and carefully planning the charging process in such a way that it introduces minimal interference and constraints on our daily schedule is of paramount

eMall App is a platform that helps the end users to plan the charging process, by getting information about Charging Points nearby, their costs and any special offer; book a charge in a specific station, control the charging process and get notified when the charge is completed. It also handles payments for the service.

In the e-Charging ecosystem, there are many different actors involved that we need to keep into consideration while collecting requirements and designing the system. The first information to consider is that Charging Points are owned and managed by Charging Point Operators (CPOs) and each CPO has its own IT infrastructure, managed via a Charge Point Management System (CPMS).

In order to comunicate with each actor, the OCPI (Open Charge Point Interface) protocol is used.

1.2.1 World Phenomena

As this system will act as a middleman between the users and the various Charging Point CPMSs, we thought that splitting the phenomena between "user side" and "CPMS side" can help to better understand.

User side

WP1	User decides to charge the electric vehicle
WP2	User goes to the Charging Point
WP3	User connects the Electric vehicle to the Charging Slot
WP4	User disconnects the Electric Vehicle from the Charging Slot

CPMS side

WP5	Charging Point starts to provide energy to the Electric Vehicle of the User
WP6	Charging Point ends to provide energy to the Electric Vehicle of the User
WP7	CPO starts the maintenance of a Charging Slot
WP8	CPO completes the maintenance of a Charging Slot
WP11	CPO update pricing policies (e.g. new special offer)
WP10	CPO update Charging Points information (e.g. new Charging Point)

Payment Service Provider side

WP9	Payment provider charges successfully the payment method registered by the user
WP10	Payment provider fails to charge the payment method registered by the user

1.2.2 Shared Phenomena

User side

SP1	User registers an account
SP2	User verifies the email for his account
SP3	User add payments information for his account
SP4	System shows the nearby available charging points to the user
SP5	User books a charging session through the system
SP6	System send information about a charging session to the User
SP7	User starts the charging session through the system

CPMS side

SP8	System books the charging session for the user via the CPMS
SP9	CPMS send to the System the charging session details
SP10	System authenticate the charging session for the CPMS
SP13	CPMS sends pricing information to the System
SP14	CPMS sends Charging Points information to the System

Payment Service Provider side

SP11	System send cost information to the Payment Provider to charge the User
SP12	Payment Provider send to the system the payment process details (eg. status)

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Connector Charging Socket	Physical connector that allow to transfer energy to the connected vehicle
Charging slot	Physical device with multiple Connectors that can charge electric vehicles. NOTE: from OCPI definitions a Charging Slot can have up to one connector active at any time (i.e. can charge only one Vehicle at any time)
Charging Point	Physical structure composed by multiple Charging Slots
Maintenance of a charging slot	Activity/activities that results in a momentary unavailability of the charging slot
Payment information	information required by the payment provider to be able to charge the user for the service (e.g. credit card number)
Charging session	period of time when the vehicle is connected to a charging plug for charging
Booking period	period of time between the booking of a charging session and the beginning of the charging session
Guest Guest User	Unregistered user
User Enabled User Active User	Registered user with confirmed email and payment method
Unconfirmed User	Registered user without confirmed email
Pending User	Registered user with confirmed email but no payment method set up
Payment Service Provider	External service that provides API to process payments

1.3.2 Acronyms

eMSP	e-Mobility Service Provider
CP	Charge Point / Charging Point
CPO	Charging Point Operator
CPMS	Charging Point Management System
OCPI	Open Charge Point Interface
EV	Electric Vehicle
PSP	Payment Service Provider

1.4 Revision History

- v1.0 - 10 December 2022
- v1.1 - 19 December 2022 | fix naming and update pricing management

1.5 Related Documents

OCPI specifications document (OCPI-2.2.1.pdf)

1.6 Document structure

The document is structured in six sections:

1. Problem introduction, associated goals of the project. In this section you can also find the scope of the project, the various phenomena occurring and the definitions and abbreviations used in this document.
2. The second section contains a overview of the systems, the details about the users types and the main functions. Class diagrams, state-charts and domain assumptions are used to introduce the various scenarios.
3. In this section are specified the different requirements of the system, the various use cases and the mapping between functional requirements and the goals of the system.
4. Alloy is used to conduct the formal analysis of the system.
5. Total effort
6. References used

2 Overall description

2.1 Product perspective

2.1.1 Scenarios

1. **User wants to start using the platform**
   Bob has bought a new electric car and wants to start using the eMall network to charge it. Bob goes to the app store to download the official application, opens it and taps on the "Signup" button to initiate the user registration flow. He inserts all the necessary information and, after submitting the form, he's granted access to the system.

2. **User wants to check for charging points availabilities**
   Bob wants to charge his car, he opens the eMall app and, using the integrated map view starts looking for a nearby CPs. After tapping on one of the available Charging Points, the application displays the status of the point, the available charging slots and their connectors as well as the price rates applied.

3. **User wants to book a charge for a certain time frame**
   Bob opens the app, searches for an available CP and, after opening the details page, he taps on the "book" button to start the booking process. The application asks for a time frame and a connector type, and after checking for availability, it books the charging slot and returns the booking reference to Bob.

4. **User wants to start a charge at the CP**
   Bob goes to a charging point and wants to start charging his vehicle, with a booking for that time frame, he simply needs to park the car in the correct spot, connect it with the correct plug and enable the charge flow via the app. Without a booking, he needs to look for an available spot via the app and start a booking process for it.

5. **User wants to be notified at the end of the charge**
   Bob started the charging process a couple of hours ago for a 2 hours time frame. The system checks the booking time-frame and sends a notification to Bob's smartphone with the details of the charging process (e.g. cost and battery status)

6. **User wants to pay for the service**
   Bob goes to retrieve his car from the CP, he uses the app to complete the charging process and the system automatically bills his credit card / Paypal account with the required amount.

2.1.2 Class Diagram

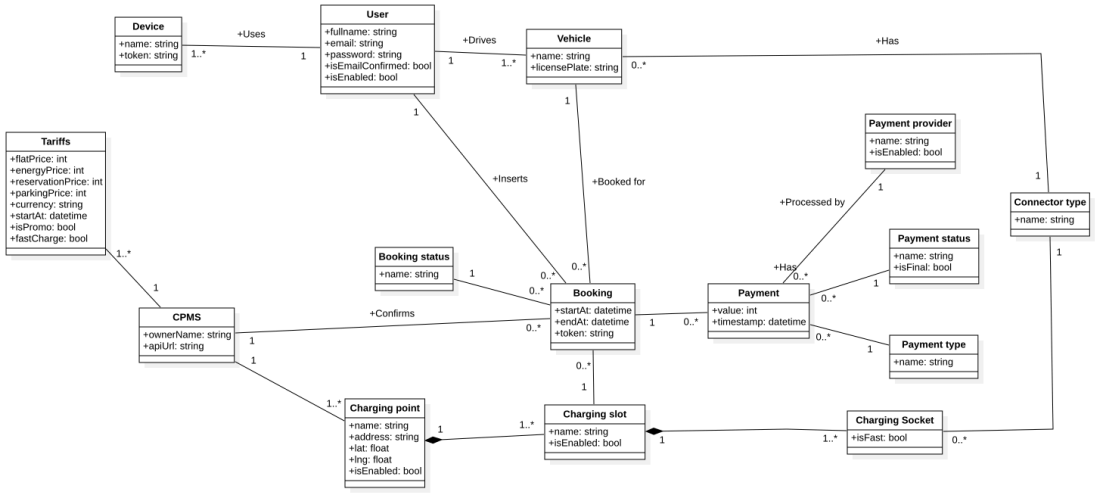

Figure 2.1: Class diagram of the eMSP system for eMall

2.1.3 State charts

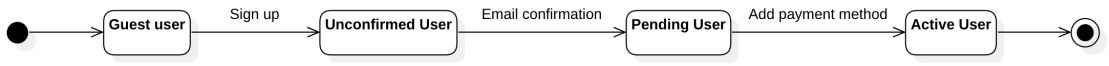

Figure 2.2: State diagram for user registration flow

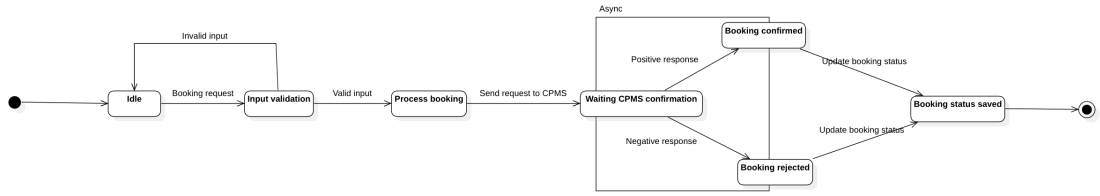

Figure 2.3: State diagram for Booking

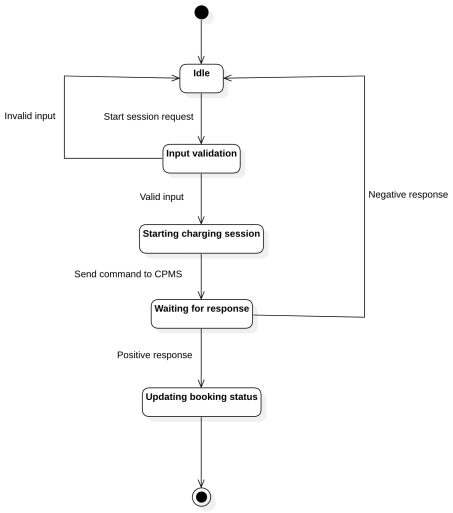

Figure 2.4: State diagram for a Charging Session

2.2 Product functions

Before going deeply into details about the product functions that needs to be developed and the related requirements, we first need to quickly introduce two base requirements needed to contain the scope of this project:

- As the system acts as a middleware between the User and the Charging Points, a structured communication protocol is needed. This system will be design to communicate using the Open Charge Point Interface (OCPI) standard.
- As OCPI enables a very complex pricing system, to keep the scope of the project simple, the system will use only the per-KW price (also called energy based price), with differentiation between fast charging and non-fast charging. Time limited promotions are also allowed. The prices are synced by the CPMSs via the PUSH interface defined in the OCPI protocol. (i.e. at any time, each CPMS can define only two active prices: standard and fast-charging)
- Another limitation that we need to impose to reduce the complexity is the various connector types. We define only two generic types of connector: standard and fast-charging, each type has its own pricing defined by the CPMS.

2.2.1 Nearby charging points overview

The users want to quickly find nearby charging points and see their status, the available slots, their connectors and the tariffs, this will allow them to decide where to charge their vehicles or not.
To achieve this, the system should act as an aggregator for multiple CPMS, collect the data in real-time and display all the information on a map.

2.2.2 Booking a charging slot

When a user has decided where to charge his vehicle, he can use the same system to book the slot for a specific time frame.
As for the OCPI standard booking is managed by the CPMS, the system will need to require booking confirmation from the CPMS.
Also OCPI standard allows only reservations that starts at the time of the request so only those type of booking will be considered in the project.

2.2.3 Unlocking a charging slot and starting a charge session

When a user arrives at the charging point he can use the app to review the booking details and check to which slot to connect the vehicle. He connects the car via the correct plug provided by the charging slot and, afterward, he uses the app to authenticate into the system and start charging. Even if OCPI standard allows for other types of session initialisation, our system will not deal with them.

2.2.4 Checking the status of the charge session

After initiating the charge the user will be able to view the status of the session (charge level and estimated time remaining) from the app.

The system automatically retrieves the needed information from the CPMS and eventually notifies the user about any problem that can occur (e.g. charge session is interrupted by the Charge Point)

2.2.5 Completing the charge session and paying for the service

A user can complete the charge session when the booked time frame is passed or even before the end of the charge.
To complete a charge session the user can use the app to stop the charging process and detach the plug from the vehicle.
As stated above, the payment request is automatically triggered and the payment service provider will charge the user the right amount and send the invoice via email.

2.3 User characteristics

For the scope of the document, we can identify a single user type for the eMall system, but to allow us to better define some details later on, we've split the user based on his possible states

1. **Unregistered user / guest**
   A person who is not already registered to the system
2. **Registered user waiting for email verification**
   A person who is registered to the system but has yet to confirm the email address used in the signup form.
3. **Registered user waiting for payment method registration**
   A person who is registered to the system, has his email verified but needs to add a payment method to start using the app.
4. **eMall active user**
   A person who is registered to the system, has a verified email address and an active and valid payment method. Payment method verification is handled by the Payment Service Provider.
5. **eMall disabled user**
   An active user that has been flagged for deactivation by the system admin. Could be for security reasons, for direct request from the customer or for terms of service violation.

2.4 Assumptions, dependencies and constraints

2.4.1 Assumptions

D1	There is a system admin that does the initial setup of the system and manages it
D2	System admin adds information about CPMS to the database
D3	Users accept the contract and terms of service when signing up into the system
D4	Users always complete their booked charging sessions
D5	PSPs always process successfully the payments

3 Specific Requirements

3.1 Use Case Diagrams

Here are displayed the most important use case diagrams for the various actors.

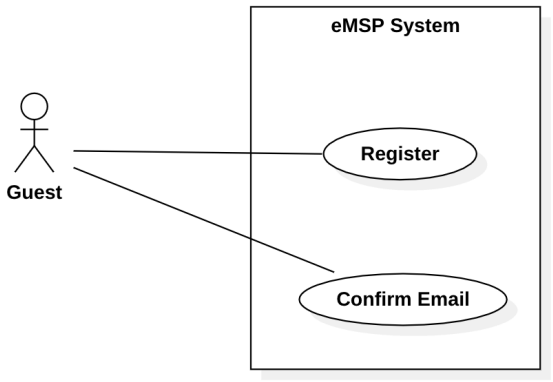

Figure 3.1: Use case diagram for guest user

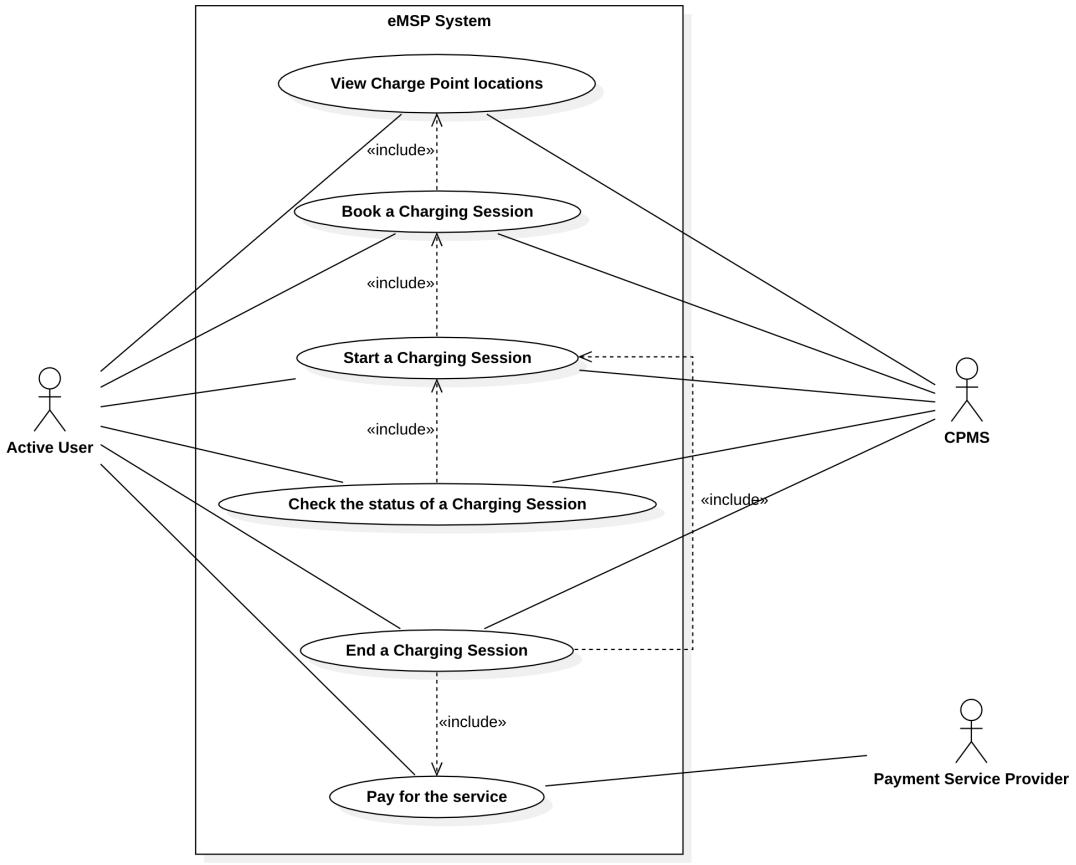

Figure 3.2: Use case diagram for an active user and the CPMS

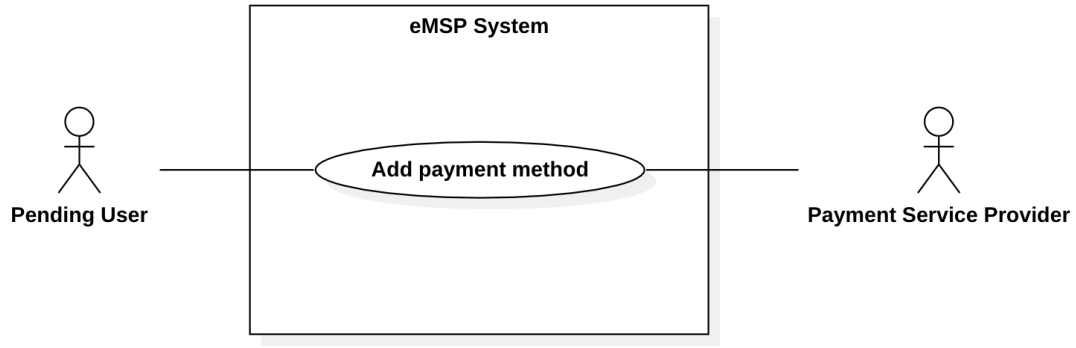

Figure 3.3: Use case diagram for the Payment Service Provider and a Pending User

3.2 Use Cases

1. **User registration**

Actor	Guest user
Entry conditions	The user is not registered to the system and is on the signup view of the mobile app
Event flow	1. The guest taps on the "Signup" button 2. The guest fills up the form 3. The guest taps on the "Submit" button 4. The system validates the data and create the user profile 5. The system shows a success message and redirects the guest to the login page
Exit condition	The user profile has been created
Exceptions	1. The guest does not insert all the required information 2. The system is unavailable 3. The system cannot process the data

2. **User login**

Actor	Registered user
Entry conditions	The user is registered into the system but has to log in first
Event flow	1. The user taps on the "Login" button 2. The user fills up the form 3. The user taps on the "Submit" button 4. The system validates the data and grants access to the user 5. The system redirects the user to the main page
Exit condition	The user is logged in
Exceptions	1. The form data are incomplete or not valid 2. The system is unavailable 3. The system cannot process the data

3. **View Charge Point locations**

Actor	Active user
Entry conditions	The user is logged in
Event flow	1. The user taps on the "Explore" button 2. The system shows a map component centred on the user location 3. The system shows all the near Charge Point locations 4. The user taps on a single location 5. The system shows the details of that location to the user (e.g. availability, connector types...)
Exit condition	The user goes to another section or closes the app
Exceptions	1. The system is unavailable

4. **Book a Charging session**

Actor	Active user
Entry conditions	The user is logged in
Event flow	1. The user taps on the "Explore" button 2. The system shows a map component centred on the user location 3. The system shows all the near Charge Point locations 4. The user taps on a single location 5. The system shows the details of that location to the user (e.g. availability, connector types...) 6. The user taps on the "Book" button 7. The system checks the availability with the CPMS 8. The CPMS confirms the booking to the system 9. The system shows a confirmation message to the user with the details of the reservation
Exit condition	The user goes to another section or closes the app
Exceptions	1. The system is unavailable 2. The CPMS is unavailable or unreachable 3. The CPMS reject the booking request

5. **Start a Charging session**

Actor	Active user
Entry conditions	The user is logged in, has an active booking and is physically at the Charging Point
Event flow	1. The user taps on the "Start Charging" button 2. The system sends the command to the CPMS 3. The CPMS confirms the start of the charging session 4. The system shows a confirmation message to the user
Exit condition	The user goes to another section or closes the app
Exceptions	1. The system is unavailable 2. The CPMS is unavailable or unreachable 3. The CPMS reject the request

6. **User is notified about the status of the Charging session**

Actor	Active user and CPMS
Entry conditions	The user is logged in, has an active session and his device is connected to the internet
Event flow	1. The CPMS sends a status update message to the system, informing that the Charging Session is completed 2. The system sends a notification via the application to the user 3. The user receives the notification
Exit condition	A notification has been sent
Exceptions	1. The system is unavailable 2. The CPMS is unavailable or unreachable 3. The CPMS reject the request 4. The user device is offline 5. The user has not allowed the app to display notifications

7. **Complete a Charging session and pay for the service**

Actor	Active user, CPMS, PSP
Entry conditions	The user is logged in, has an active session and is physically at the Charing Point
Event flow	1. The user taps on the "End Charging" button 2. The system sends the command to the CPMS 3. The CPMS confirms the end of the charging session and returns the applied tariff 4. The system shows a success message to the user 5. In the background the system send a request to the PSP to charge the user for the service
Exit condition	A notification has been sent
Exceptions	1. The system is unavailable 2. The CPMS is unavailable or unreachable 3. The CPMS reject the request 4. PSP is unavailable 5. PSP cannot process the payment

3.3 Sequence Diagrams

Here are presented the sequence diagrams for the most important use cases. Only the "success" flow is displayed, every exception in the communication flow results in an error message displayed to the user and a request retry between the eMSP and both a CPMS or a PSP with random time with a maximum number of retries based on the request type.

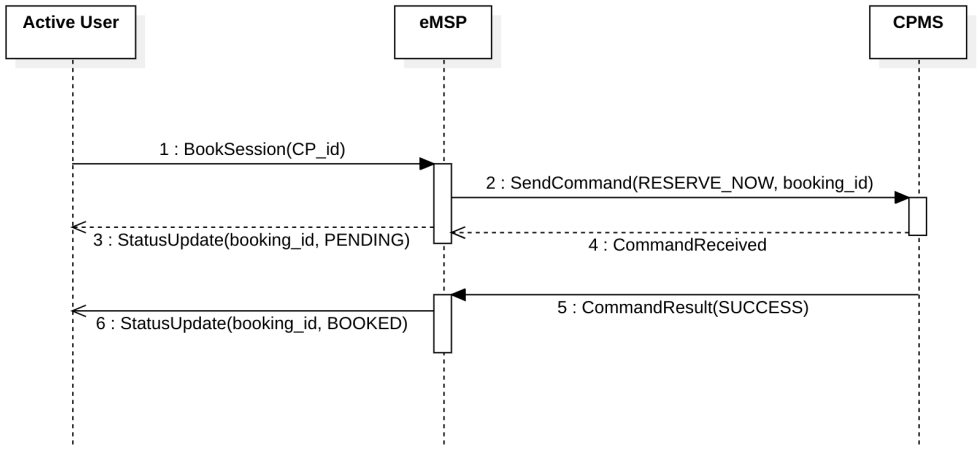

Figure 3.4: Sequence diagram for the booking process of a new charging session

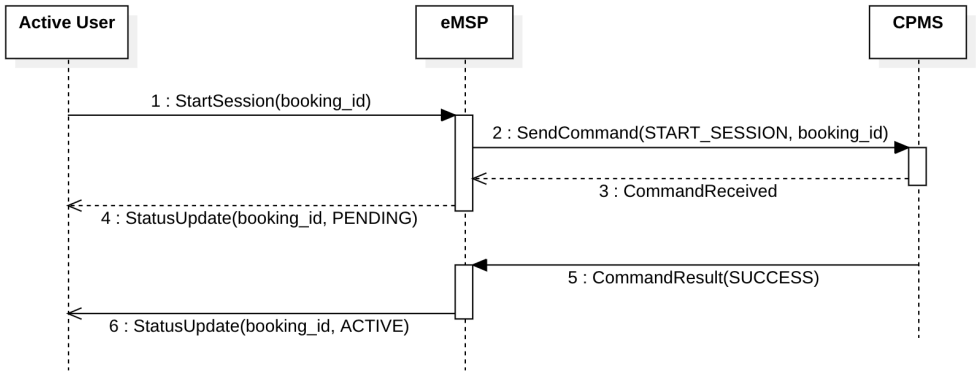

Figure 3.5: Sequence diagram for the starting of a charging session

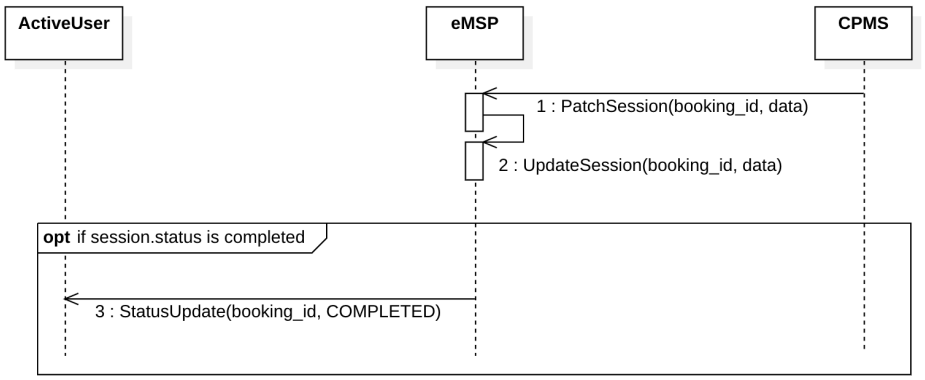

Figure 3.6: Sequence diagram for the notification of a relevant update in a charging session

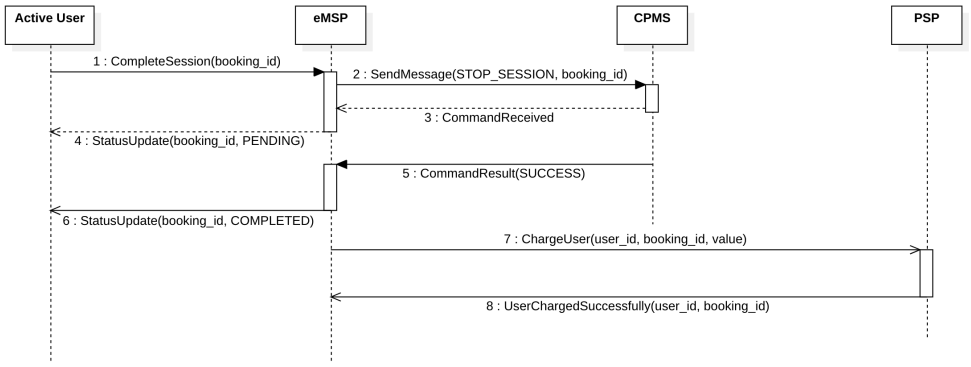

Figure 3.7: Sequence diagram for the completion of a charging session and the related payment

3.4 Functional Requirements

The system should:

Requirement	Description
R1	Allow Guest User sign-up
R2	Allow Login for Active User
R3	Require verification for email
R4	Require verification for payment method
R5	Allow to get a list of Charge Point
R6	Allow to book a charging session on a specific Charge Point
R7	Allow to start a charging session for a specific booking
R8	Allow to view the list of bookings
R9	Allow to complete a charging session
R10	Forward the payment request to the PSP
R11	Allow to get a list of Charge Point based on the location
R12	Allow to filter the list of Charge Point based on the location
R13	Be able to send push notifications to the user device
R14	Periodically poll the various CPMS to cache the data

3.4.1 Requirements mapping on Goals

Goal	Requirements
G1	R1, R2, R3, R4, R5, R11, R12, R14
G2	R1, R2, R3, R4, R5, R6, R8, R11, R12, R14
G3	R1, R2, R5, R7, R7, R8, R11, R12, R14
G4	R1, R2, R5, R6, R7, R8, R11, R12, R14
G5	R1, R2, R5, R6, R7, R8, R11, R12, R13, R14
G6	R1, R2, R4, R9, R10, R14

3.5 Performance Requirements

As most of the Charge Points are located in the city centre, where internet connectivity is usually optimal and at high speed, the system does not have special requirements for performances related to bandwidth usage. Also, no particular usage spikes are expected at any moment, so server load should be directly connected to the number of users and Charge Points covered by the system.

3.6 Design Constraints

3.6.1 Standards compliance

There are two legal requirements to follow: GDPR for sensible user data and PS2 / KnowYourCustomer for the payment processing. The second requirement can be fulfilled by selecting a compliant PSP. Finally, to enable the interconnection with the different CPMS, the OCPI standard shall be followed when implementing the communication protocol and the data structures.

3.6.2 Hardware limitations

No particular hardware are required to use the System, other than an internet-capable smartphone.

3.7 Non-functional Requirements

NFR1	Reliability impacts the system usage and the related business, to keep it as high as possible, fall back servers shall be considered while designing the infrastructure.
NFR2	Availability also has a huge impact but as this is not a sensible service that needs to be up and running at all time, a good compromise with cost effectiveness can be considered.
NFR3	The basic security standard "de-facto" measurements should be followed, SSL and HTTPS should be enough to offer both security and ease to use.

3.8 External Interface Requirements

3.8.1 User Interfaces

The user interface for the eMSP system is a mobile application available for both iOS and Android devices.
Also, as this will be a publicly available application, accessibility should be considered a priority when designing the interface.

3.8.2 Hardware interfaces

The only hardware required to use the system is a Smartphone running iOS or Android OS, with internet connection capabilities and GPS functionalities.

3.8.3 Communication interfaces

As the system needs to communicate with multiple actors, the following communication interfaces are required:

1. **CPMS communication:**
   To share data with the various CPMS, an active, low-latency internet connection is needed. Also is required that the CPMS follows the OCPI protocol.
   Is important to remember that to follow the OCPI protocol, our system is required to expose a "PUSH interface" (e.g. an API endpoint that follows the protocol structure) that enables the CPMS to send newer data in real-time.

2. **Payment Service Providers:**
   To process payments a connection with a Payment Service Provider is needed. This connection happens via proprietary API, so custom components are required to connect different providers.

4 Formal analysis

4.1 Alloy formal analysis

In this section is described a formal analysis written using Alloy. The purpose of the analysis is to highlight the key model relationships and the main constraints of the system.

The model has been built following the class diagram, the included code contains all the models and the relationships, and it shows that the resulting system is feasible and useful and it justifies our modelling decisions.

In the following sections you will find the code used to run the analysis and some of the world that the Alloy tools generated from our definitions.

4.2 Alloy code

```
//FILE: alloy_eMSP.als

//abstract sig String {} -- commented as the used version of Alloy includes it
abstract sig Bool {}
one sig TRUE extends Bool {}
one sig FALSE extends Bool {}


sig ConnectorType {}
sig Device {} {
   (one u: User |this in u.devices) and -- device needs a user
   (no disj u, u': User |this in u.devices & u'.devices) -- no device can be owned by
       ↪ multiple users
}

sig LicensePlate {} {
   (one v: Vehicle |v.licensePlate =this) -- licence is used only as string here so
       ↪ cannot exists without a vehicle
}
sig Vehicle {
   licensePlate: one LicensePlate,
   connector: one ConnectorType
} {
   (one u: User |this in u.vehicles) and -- vehicle needs a user
   (no disj u, u': User |this in u.vehicles & u'.vehicles) -- no vehicle can be owned by
       ↪ multiple users
}

sig User {
   devices: set Device,
   vehicles: set Vehicle,
   bookings: set Booking
} {
   #devices >0 and #vehicles >0
}


sig CPMS {
   chargingPoints: set ChargingPoint,
   tariffs: set Tariff,
} {
   (#chargingPoints) >0 and
   (#tariffs >0)
}

sig ChargingSocket {
   isFast: one Bool,
   connectorType: one ConnectorType
} {
```

```
   (one slot: ChargingSlot |this in slot.sockets) and -- sockets needs to have one slot
       ↪ connected
   (no disj s, s': ChargingSlot |this in s.sockets & s'.sockets) -- no multiple slots
       ↪ for the same socket
}

sig ChargingSlot {
   sockets: set ChargingSocket
} {
   (one cp: ChargingPoint |this in cp.chargingSlots) and -- charging slot needs to be in
       ↪ a charging point
   (no disj cp, cp': ChargingPoint |this in cp.chargingSlots & cp'.chargingSlots) and --
       ↪ a charging slot cannot be in multiple charging points
   (#sockets >0)
}

sig ChargingPoint {
   chargingSlots: set ChargingSlot
} {
    (one cpms: CPMS |this in cpms.chargingPoints) and -- charging point needs a CPMS
    (no disj cpms, cpms': CPMS |this in cpms.chargingPoints & cpms'.chargingPoints) and
        ↪ -- charging point cannot have multiple CPMS
    (#chargingSlots >0)
}

sig Tariff {
   startAt: one Int,
   fastCharge: one Bool,
} {
   (startAt >0) and -- dates as timestamps so they need to be > 0
   (one cpms: CPMS |this in cpms.tariffs) and -- tariff needs a CPMS
   (no disj c, c': CPMS |this in c.tariffs & c'.tariffs) -- no tariff can be used for
       ↪ multiple CPMS
}

sig BookingStatus {

}

sig Booking {
   status: one BookingStatus,
   vehicle: one Vehicle,
   slot: one ChargingSlot,
   payments: set Payment
} {
   (one user: User |this in user.bookings and vehicle in user.vehicles) and -- booking
       ↪ needs a user and vehicle must be owned by the same user
   (no disj u, u': User |this in u.bookings & u'.bookings) and
   (no disj p, p': payments |p.status.isFinal =TRUE and p'.status.isFinal =TRUE) --
       ↪ only one successful payment must happen for a single booking
}

sig PaymentStatus {
```

```
    isFinal: one Bool
}
sig PaymentType {}

sig PaymentProvider {
    payments: set Payment
}

sig Payment {
    timestamp: one Int,
    status: PaymentStatus,
    type: PaymentType
} {
    (timestamp >0) and
    (one provider: PaymentProvider |this in provider.payments) and -- payment needs a
        ↪ provider
    (no disj p, p': PaymentProvider |this in p.payments & p'.payments) and -- only one
        ↪ provider per payment
    (one booking: Booking |this in booking.payments) and -- payment needs a booking
    (no disj b, b': Booking |this in b.payments & b'.payments) -- only one booking for
        ↪ payment
}

/** ---- FACTS ---- */
fact noDifferentVehiclesWithSameLicensePlate {
    no disj v, v': Vehicle |
        v.licensePlate =v'.licensePlate
}

fact noPaymentsAfterASuccessfulOne {
    no disj p, p': Payment |(
        p.timestamp <p'.timestamp
        and p.status.isFinal =TRUE
    )
}

fact noMultiplePricingActiveAtTheSameTimeForTheSameCPMS {
    all cpms: CPMS |(
        no disj t, t': cpms.tariffs |(
            t.startAt =t'.startAt and
            t.fastCharge =t'.fastCharge
        )
    )
}

/** ---- ASSERTIONS ---- */
assert deviceHasOneUser {
    no d: Device, u, u': User |
        u ≠u' and
        d in u.devices and d in u'.devices
    and
    all d: Device |
        one u: User |d in u.devices
```

```
}

check deviceHasOneUser

assert bookingHasOnlyOnePaymentWithFinalStatus {
   all b: Booking |
      no p, p': Payment |(p ≠p' and p in b.payments and p' in b.payments and
         p.status.isFinal =TRUE and p'.status.isFinal =TRUE
      )
}

check bookingHasOnlyOnePaymentWithFinalStatus

assert noMultiplePricingActiveAtTheSameTimeForTheSameCPMS {
   no disj t, t': Tariff |(
      t.~tariffs =t'.~tariffs and t.startAt =t'.startAt and t.fastCharge =t'.fastCharge
   )
}

check noMultiplePricingActiveAtTheSameTimeForTheSameCPMS

/** ---- PREDICATES ---- */

pred noCPMSInsertedWorld {
   #User >0 and
   #CPMS =0
}
run noCPMSInsertedWorld for 5

pred world1 {
   #User =1
   #Payment > 0
}
run world1 for 5

pred world2 {
   #User >1 and
   #CPMS >1 and
   #Booking >1 and
   #Payment > 1 and
   (some b: Booking |#b.payments >1 and one p: b.payments |p.status.isFinal =TRUE) and
   (some cpms: CPMS |#cpms.tariffs >1)
}
run world2 for 5
```

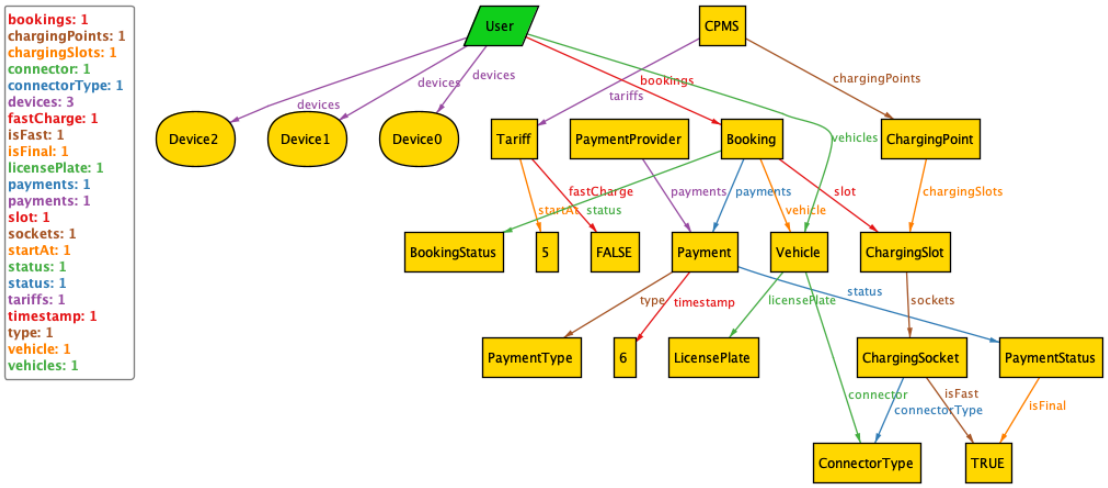

Figure 4.1: World generated by Alloy (world1) - Basic world with one user

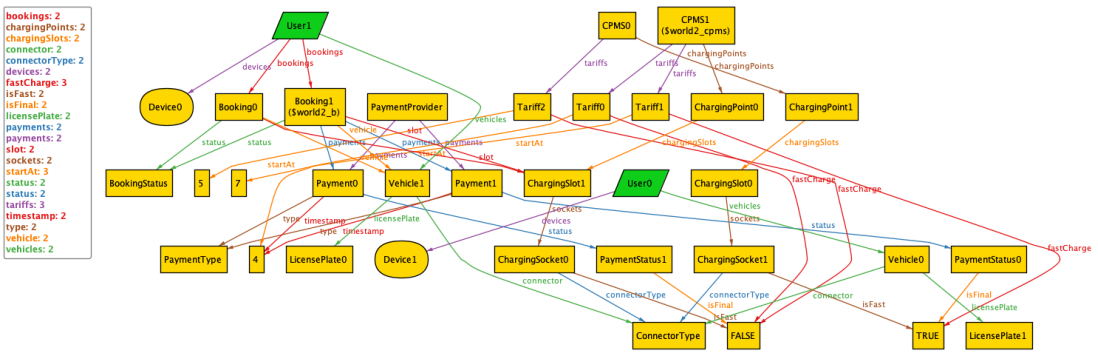

Figure 4.2: World generated by Alloy (world2) - Advanced world with multiple CPMS