POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

eMall – e-Mobility for All

DD
Software Engineering 2

TEAM MEMBERS:

Andrea Piras - 10725972
Emanuele Santoro - 10676582
Andrea Sanguineti - 10739788

Academic Year: 2022-23

Contents

Contents **i**

1 INTRODUCTION **1**
1.1 Purpose . . . 1
1.2 Scope . . . 2
1.3 Definitions, Acronyms, Abbreviations . . . 2
1.3.1 Definitions . . . 2
1.3.2 Acronyms . . . 3
1.3.3 Abbreviations . . . 3
1.4 Document Structure . . . 4

2 ARCHITECTURAL DESIGN **5**
2.1 Overview . . . 5
2.2 Component View . . . 6
2.3 Deployment View . . . 12
2.4 Runtime View . . . 13
2.4.1 Sequence Diagrams . . . 13
2.5 Component Interfaces . . . 31
2.6 Architectural Styles and Patterns . . . 32
2.6.1 Architectural Styles . . . 32
2.6.2 Patterns . . . 32
2.7 Other Design Decisions . . . 33

3 USER INTERFACE DESIGN **34**
3.1 eMSP Application . . . 34
3.2 CPMS . . . 38

4 REQUIREMENTS TRACEABILITY **45**
4.1 Explicit Mapping On Goals . . . 45
4.2 Mapping On Requirements . . . 59
4.2.1 Mobile Application . . . 59
4.2.2 eMSP Application Server . . . 60
4.2.3 CPMS . . . 62

5 IMPLEMENTATION, INTEGRATION AND TEST PLAN **64**
5.1 Implementation and Component Integration . . . 64
5.2 System Testing . . . 67

6 EFFORT SPENT **68**

7 REFERENCES **69**
7.1 Reference Documents 69
7.2 Reference Sites 69

1 | INTRODUCTION

1.1. Purpose

Electric vehicles are the key technology to reduce the environmental impact of road transport, a sector that accounts for 16% of global emissions. Recent years have seen exponential growth in the sale of electric vehicles together with improved range, wider model availability and increased performance.

This trend is projected to continue in the future, and for this reason, more charging stations are being built each year, in order to satisfy the increasing demand for energy for these electric vehicles.

In this context, it is fundamental that the communication between the charging stations and the Drivers is managed in such a way that it introduces minimal interference and constraints on our daily schedule.

eMall is going to be a platform that permits the aforementioned communication through its subsystems eMSP (e-Mobility Service Provider), which is used by some users, like Drivers, and the CPMS (Charge Point Management System), used by the owner of the charging stations or also called Charging Point Operator (CPO).

The main goal of this document is to provide an overall view of the architecture of the software product which has already been discussed in the RASD. Guidelines and design constraints about the system are shown in order to help the development team to go through the implementation, integration and testing phase in a more organized way.

1.2. Scope

The eMall platform objective is the integration between two subsystems: the eMSP and the CPMS.
Even though eMSPs could be handled directly by third-party providers, in this project both subsystems are managed by eMall.
A third-party provider can still use this document as a guideline for implementing the eMSP as long as it will respect the main external interfaces with eMall and with the CPMSs.
In this project, only the architecture of the subsystems is described, since eMall is considered an already implemented service accessible via web and therefore only mentioned if the actors or the subsystems interact with it.

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

Definition	Description
Energy Provider	The entity providing energy to the charging station, it can either be a DSO or the internal batteries of that charging station.
Driver	The person that will use the eMSP application. For shortening reasons, only male pronouns are used to address the Driver.
Charging Point Operator	The legal owner of charging stations that will use the CPMS. For shortening reasons, only male pronouns are used to address the Driver.
Charging Station	Physical place managed by a CPO where Drivers can charge their vehicle. It has one or more charging sockets. When a charging station is "available, " one of its sockets is available.
Charging Socket	Plug where Drivers connect their vehicle to charge it. It can support different charging types. When a charging socket is "available" it means that it has no booked charging process or no one is connected to it.
Charging Type	It defines the speed of the charging process.
User	Either the CPO or the Driver.

Table 1.1: Definitions

1.3.2. Acronyms

Acronyms	Description
eMall	e-Mobility for All
eMSP	e-Mobility Service Provider
CPO	Charging Point Operator
CPMS	Charge Point Management System
DSO	Distribution System Operator
FRE	Functional REquirement
RASD	Requirements Analysis and Specification Document
ID	IDentifier
API	Application Programming Interface
TDD	Test Driven Development

Table 1.2: Acronyms

1.3.3. Abbreviations

Abbreviation	Description
Gx	Goal number X
FREx	Functional Requirement number X
opt	Optional
alt	Alternative
par	Parallel
ref	Reference

Table 1.3: Abbreviations

1.4. Document Structure

Section 1 This section contains the scope the purpose of the DD document. This chapter also includes the structure of the document and a set of abbreviations, acronyms and definitions used.

Section 2 This section contains the architectural design choices, there are an overview of the designed architecture, all the components, the interfaces, and the technologies used for the design of the application. This chapter also includes the functions of the interfaces and the processes in which they are utilized, highlighting the interaction between them. At the end there is an explanation of the design pattern chosen to develop the application and any other design decisions.

Section 3 This section contains a representation of how the User Interfaces should look like. It completes the User Interfaces subsection present inside the RASD.

Section 4 This section contains a description of how the requirements defined in the RASD map to the design elements described in this document in order to satisfy the goals, which are defined in the RASD as well.

Section 5 This section contains the plan which describes how to implement the sub-components of the systems and in which order. In addiction to this there is also the test planning for the two systems.

Section 6 In this section is showed how much time every student spent while working at this document.

Section 7 This section is made in order to point out all the references and tools used during the creation of this document.

2 | ARCHITECTURAL DESIGN

2.1. Overview

eMSP The eMSP Application is going to be a distributed one and it will be divided into three software levels (tiers): presentation, logic and data. This tier division is used in order to have maintainability and scalability.

1. **Presentation:** All the components that will be used in order to provide the final user interfaces. They are provided by the Mobile Application.
2. **Logic:** All the components that contain logic for processing data. Since the Mobile Application is going to be a fat client (both presentation and logic) some of the logic components will be contained in it, meanwhile, the other components will be located on the Application Server.
3. **Data:** All the components which implement storage and management of data. This includes DBMS and databases.

The interaction between different entities will be handled by the logic tier which will include specific components able to grant these types of interactions:

- Mobile Application - eMSP Application Server
- Mobile Application - external APIs
- eMSP Application Server - CPMSs
- eMSP Application Server - external APIs

CPMS The CPMS application will be instead a monolithic application that will include all the aforementioned tiers in just one single application. A monolithic application is better than a three-tier architecture for the CPMS application due to

- **Simplicity**: Developers can easily understand the overall structure and apport small changes if requested by the CPO
- **Deployment**: The application is easy to deploy and can be contained in a single package ready to be executed by a server owned by the CPO

- **Integration**: Since the application's interfaces are well-defined and consistent it will be easier to integrate with other applications such as the eMSP.

During the development of the subsystem, it is important to consider how the eMSP and the CPMSs will communicate. When the eMSP needs to perform an operation on a specific CPMS (such as starting the charging process), it will send a request with the specific operation. Additionally, whenever there is a relevant update to charging station data, the CPMS will send the updated data to all previously associated eMSPs through a specific component. This asynchronous method of reading data for the charging stations is necessary because it allows the eMSP to store all updated data from the CPMS, improving scalability for Drivers. If the data were read synchronously, the eMSP would need to send a request to the specific CPMS every time a Driver wants to check charging station information, which could reduce scalability as the number of Drivers increases.

2.2. Component View

In this section, the component view of the two systems is presented with a high level component diagram, which also highlights external interface interaction.
In the following sections, detailed diagrams are constructed in the same manner as described in section 2.1, with three tiers for the eMSP and a monolithic approach for the CPMS.

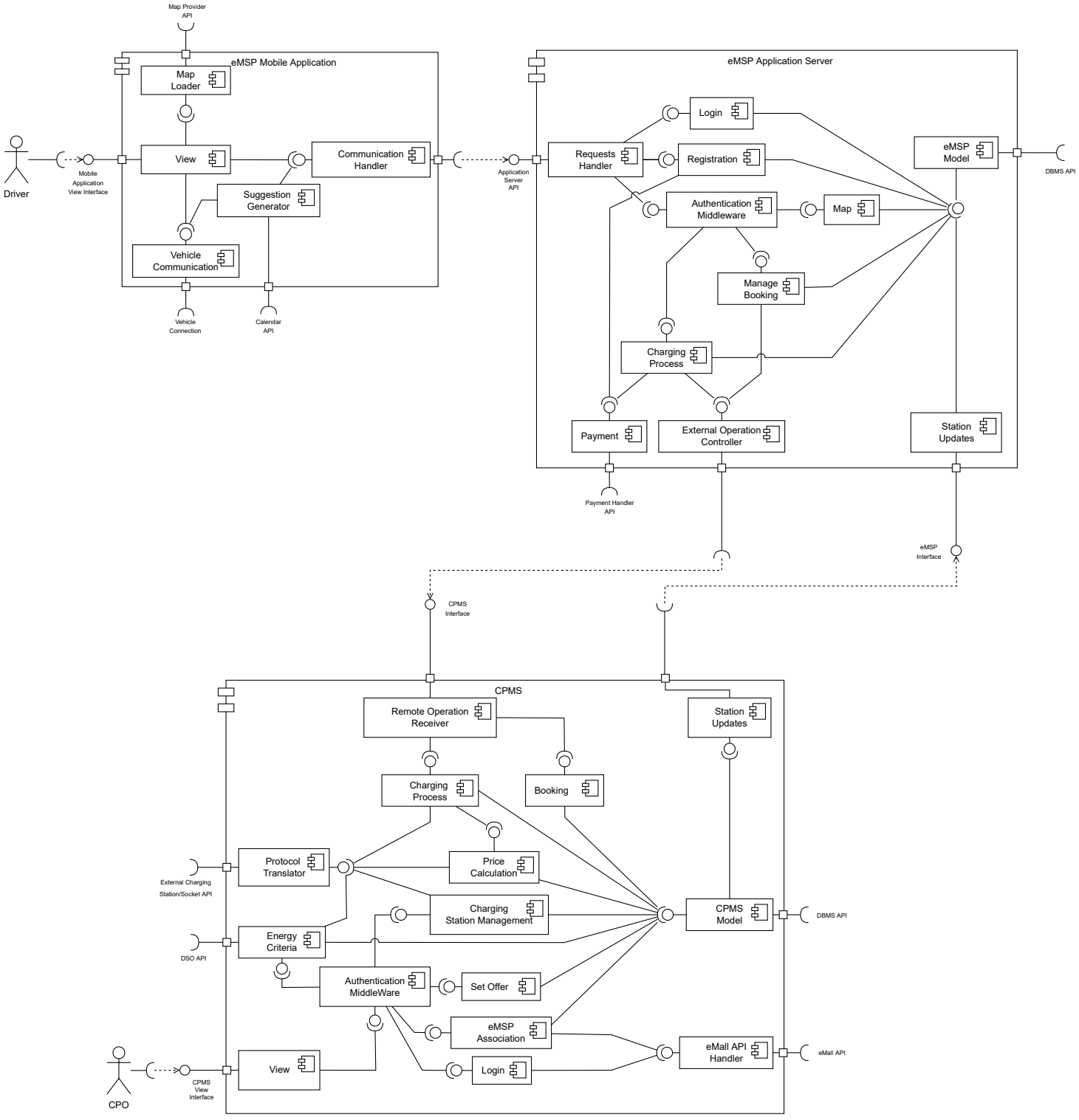

Figure 2.1: Component Diagram

eMSP Mobile Application

Presentation Level

- **View**: Component that will handle all the front-end rendering for the mobile application.

Logic Level

- **Map Loader**: Component that will handle the communication with the Maps external API.
- **Vehicle Communication**: Component that will handle the communication with the vehicle.
- **Suggestion Generator**: Component that will handle the creation of personalised suggestions for the Driver.
- **Communication Handler**: Component that will handle the communication with the eMSP Application Server.

eMSP Application Server

All components of the eMSP Application Server belong to the Logic level.

- **Requests Handler**: Component that will handle all the received requests.
- **Login**: Component that will handle the login request and authenticate a Driver.
- **Registration**: Component that will handle the registration requests and create a new account for a Driver.
- **Authentication Middleware**: Component that will handle every request checking if the Driver requesting is authenticated and forwarding the request to the respective component.
- **Map**: Component that will handle requests for the charging stations present on the map based on their location.
- **Manage Booking**: Component that will handle booking requests by creating or deleting them.
- **Charging Process**: Component that will handle requests that need to perform operations related to a charging process.
- **Station Updates**: Component that will handle requests from CPMSs in order to update eMSP Model when any data related to charging stations is updated.
- **External Operation Controller**: Components that will handle external operation that needs communication with the CPMSs in order to be executed.
- **Payment**: Component that will handle communication with external payment API in order to perform money transactions.
- **eMSP Model**: Component that will handle communication with the DBMS.

CPMS

- **View**: Component that will handle all the front-end rendering for the CPMS application.
- **Authentication Middleware**: Component that will handle every request checking if the CPO requesting is authenticated and forwarding the request to the respective component.
- **Energy Criteria**: Component that will handle the modification and update of both energy acquisition and revenue criteria.
- **Login**: Component that will handle the login request and authenticate a CPO.
- **eMSP Association**: Component that will handle the association process between the CPMS and eMSPs.
- **Set Offer**: Component that will handle the requests of setting offers by the CPO.
- **eMall API Handler**: Component that will handle the communication with the eMall API
- **Price Calculation**: Component that will calculate the price after a charging process ends.
- **Charging Station Management**: Component that will handle the requests of charging station management by the CPO.
- **Protocol Translator**: Component that will handle communication with charging stations API and charging sockets API
- **Remote Operation Receiver**: Components that will handle external requests from the eMSP.
- **Booking**: Component that will handle the booking process for a charging station.
- **Charging Process**: Component that will handle the charging process for a charging station.
- **Station Update**: Component that will perform requests to associated eMSPs in order to update them when any data related to charging stations is updated.
- **CPMS Model**: Component that will handle the communication with the DBMS.

eMSP Model ER Diagram

The following ER diagram displays a detailed view of a possible internal structure of the *eMSP Model* component.

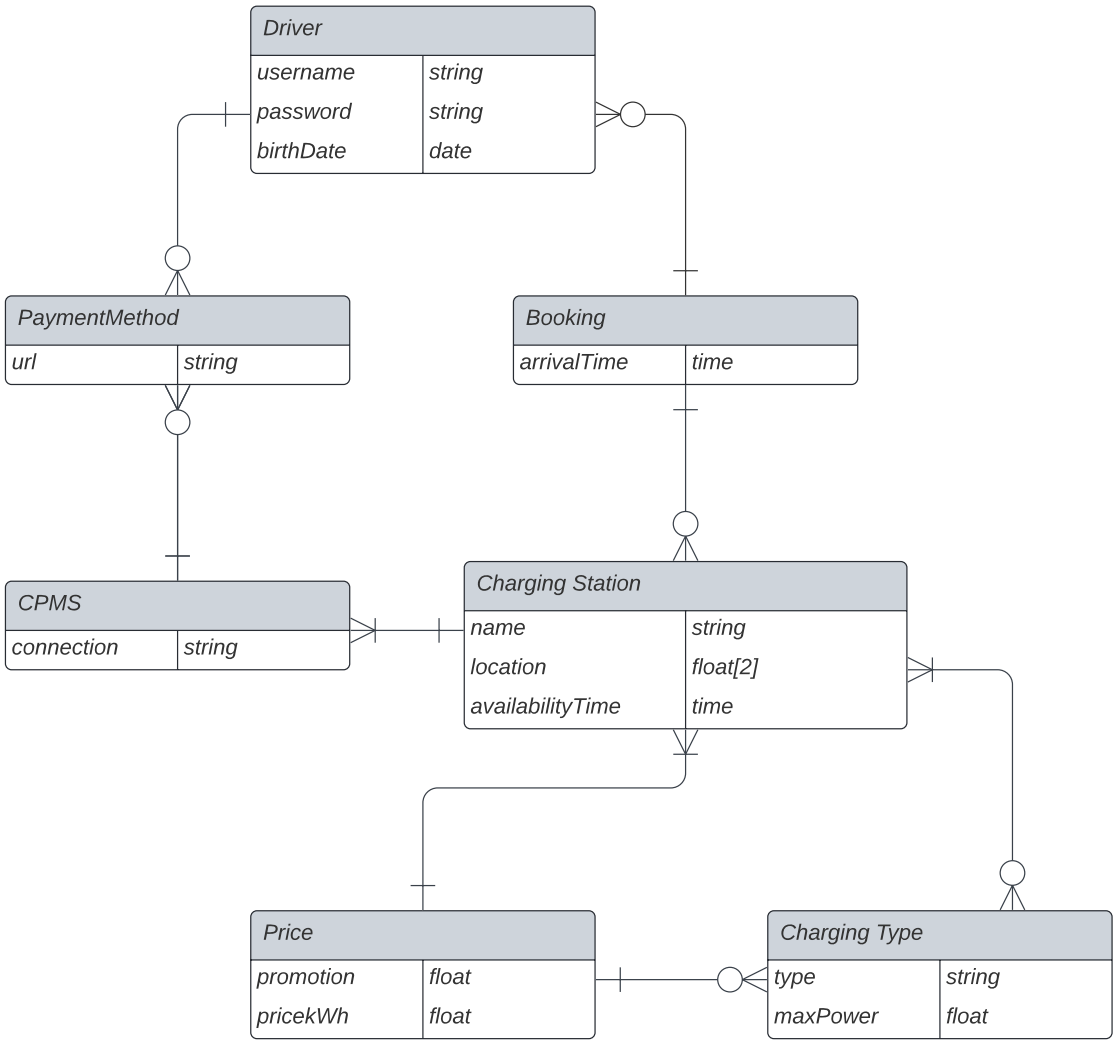

Figure 2.2: eMSP Model ER Diagram

CPMS Model ER Diagram

The following ER diagram displays a detailed view of a possible internal structure of the *CPMS Model* component.

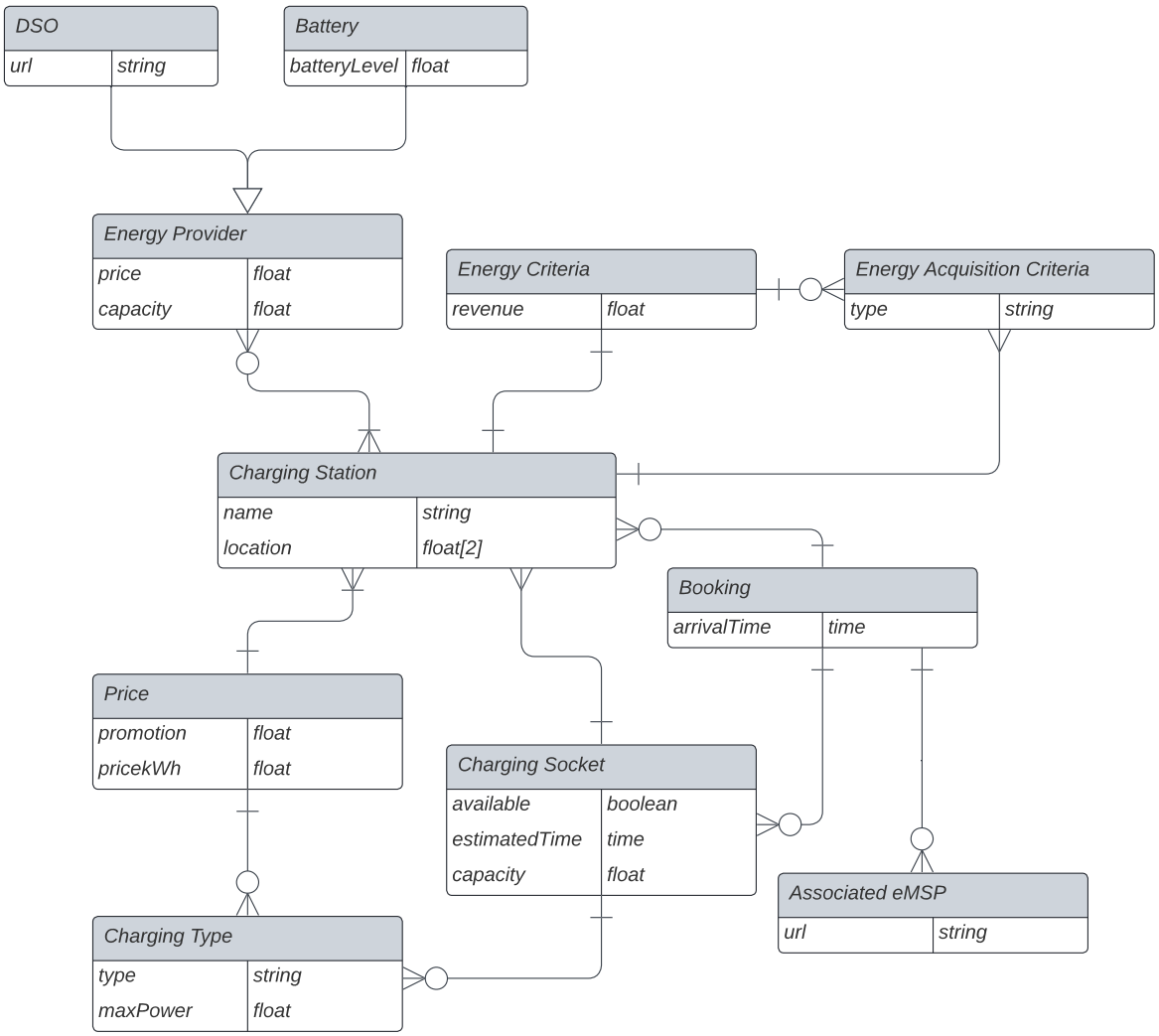

Figure 2.3: CPMS Model ER Diagram

2.3. Deployment View

The following deployment diagram shows the execution architectures of the systems. It specifies the hardware and software environments as well as middlewares present and protocols needed to communicate between the systems.

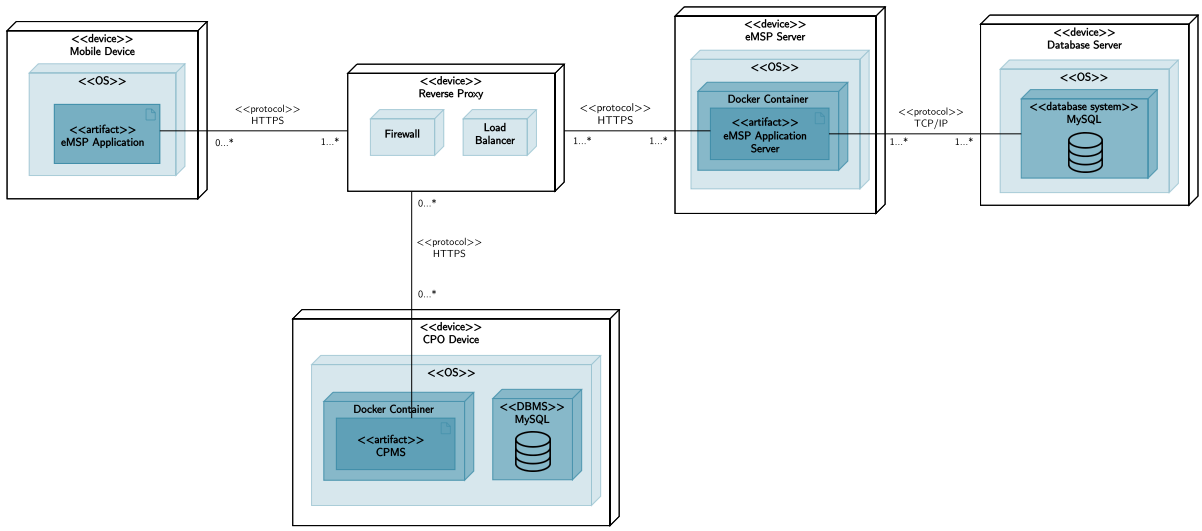

Figure 2.4: Deployment Diagram

- **Mobile Device:** Personal device with internet access used by the Driver to run the eMSP Application.
- **Reverse Proxy:** The reverse proxy acts as a single point of contact for users, protecting the application servers behind it from malicious traffic using a firewall. It also includes a load balancer, which distributes incoming requests evenly across the application servers to improve performance, availability and reliability. Multiple reverse proxies can be deployed in different locations to increase availability.
- **eMSP Server:** The application server hosts the eMSP's application logic and interacts with the database server. Using multiple application servers can improve reliability and availability and allow the system to handle a larger number of concurrent requests.
- **Database Server:** The database server, which includes a database management system (DBMS), will host the database. Using multiple database servers can improve reliability and increase the availability of the eMSP.
- **CPO Device:** The CPO uses a personal device with internet access to run the CPMS application. In order to receive external requests from eMSPs through the internet, the device must also have a firewall to protect against potential security threats.

2.4. Runtime View

The following sequence diagrams depict the behaviour of the key components of the eMSP and the CPMS subsystems. These diagrams provide a high-level overview of the interaction between the various components. Further details and implementation will be addressed during the development process.

2.4.1. Sequence Diagrams

A detail that should be noticed in the following sequence diagrams, is the async request made by the *View* (Both for the eMSP and the CPMS). This is needed in order to let the Driver have a responsive and fluid user interface.

1. **Driver Authentication Check**

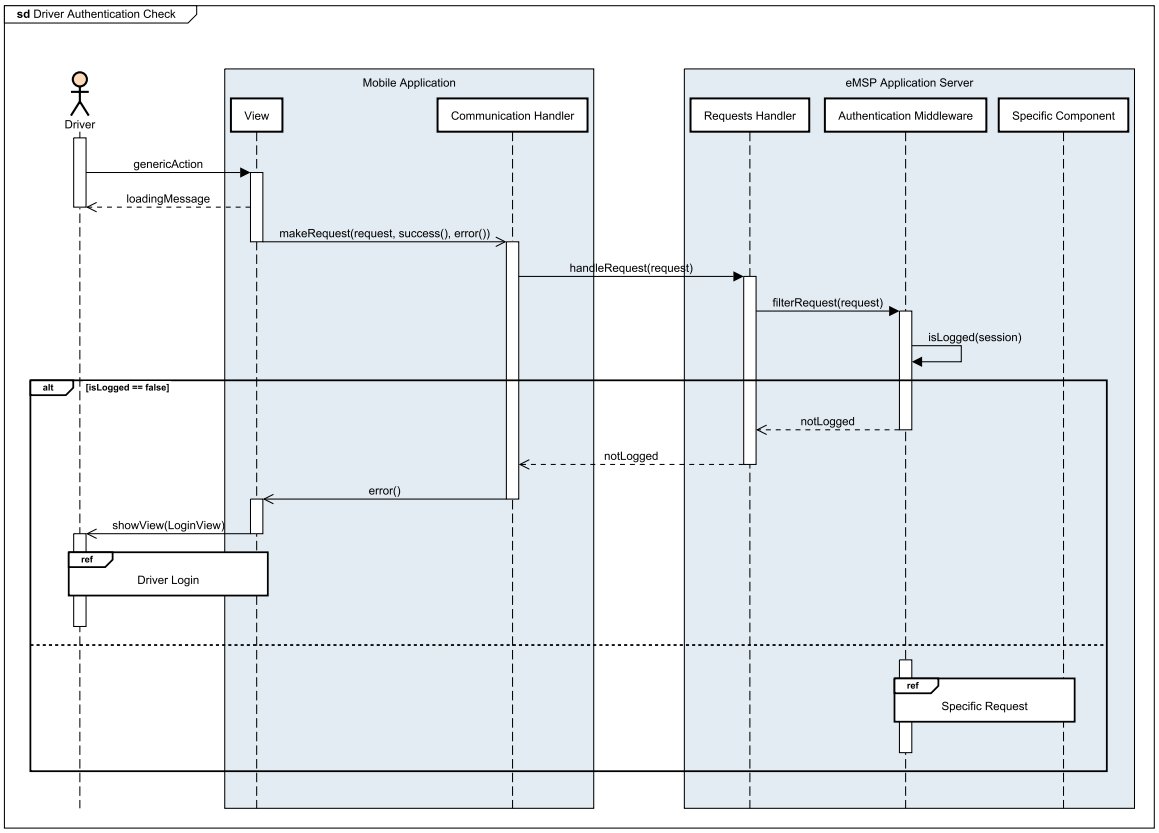

Figure 2.5: Authentication of a Driver

The figure illustrates the process of authenticating a Driver for every request, except for login and registration. The *Specific Component* in the figure refers to the component capable of responding to the Driver's request. If the Driver is not authenticated, their mobile application will be redirected to the login view. To implement the isLogged(session) function in the sequence diagram and create a session to identify the Driver, it is recommended to use a technology like JWT, which avoids the need to query the DBMS.[7.2]

2. **Driver Registration**

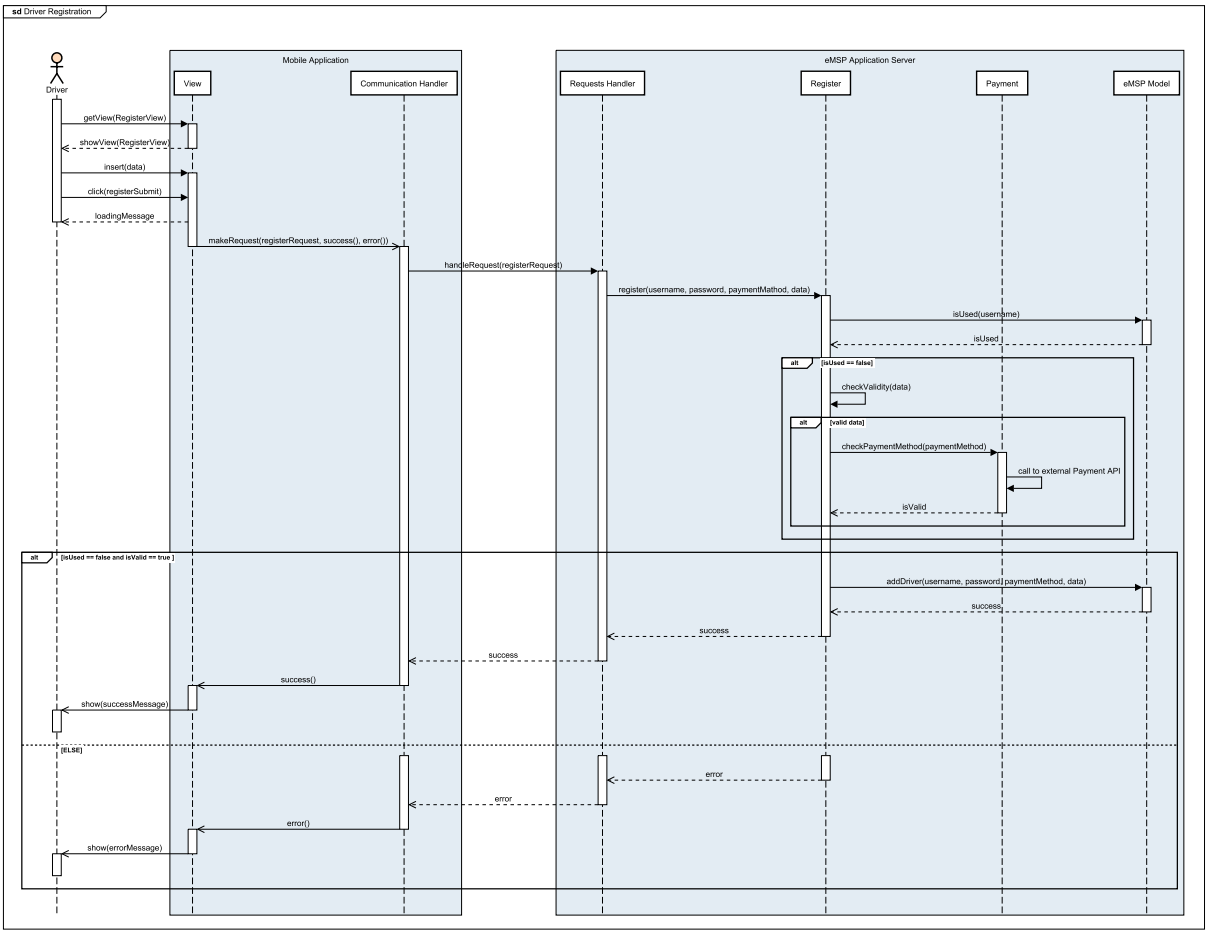

Figure 2.6: Registration of a Driver

The figure shows the process of the creation of a new account for a Driver. In particular, if the Driver wants to register himself on the platform, he should insert a valid payment method and credentials, otherwise the process will fail.

3. **Driver Login**

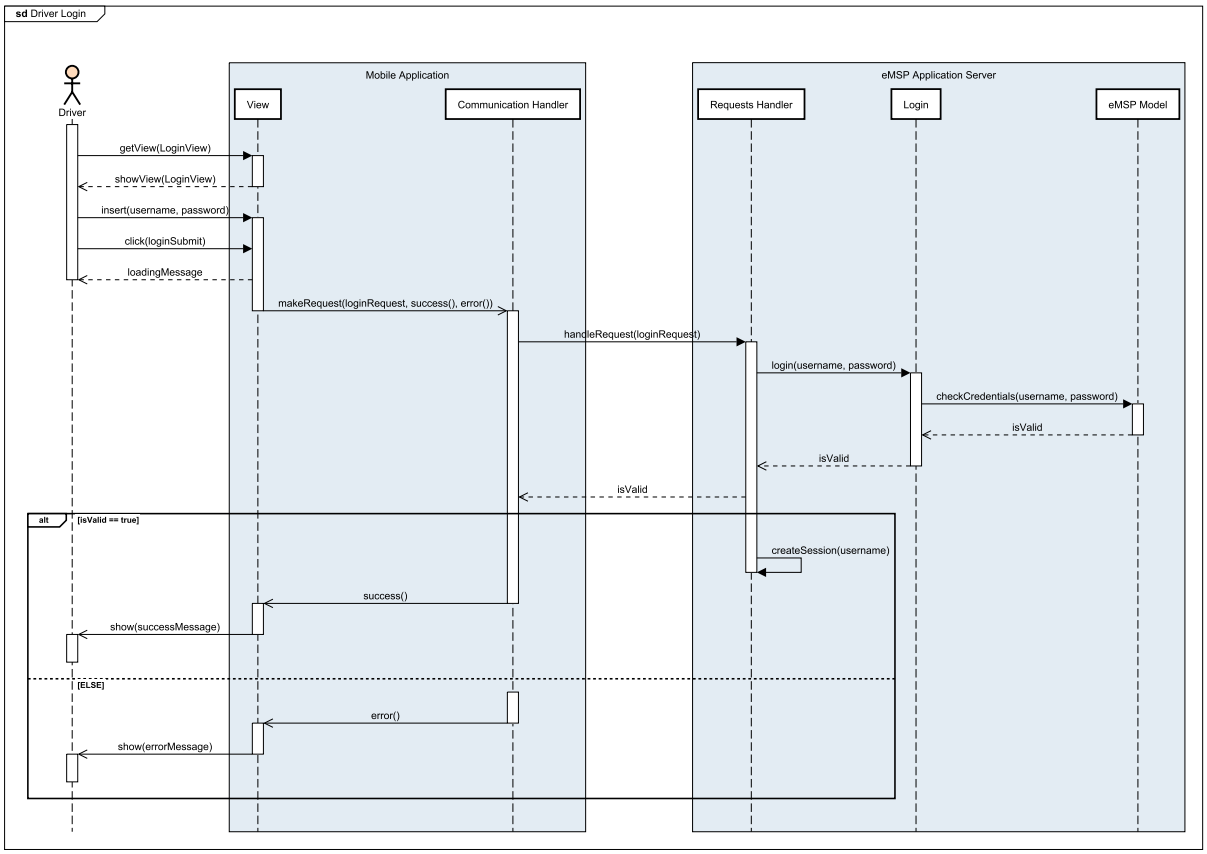

Figure 2.7: Login of a Driver

The figure shows the process of login. As aforementioned, after login, the platform should create a session in order to let the Driver stay authenticated. This session should provide basic information about the Driver such as his ID number. 7.2

4. **Check Map**

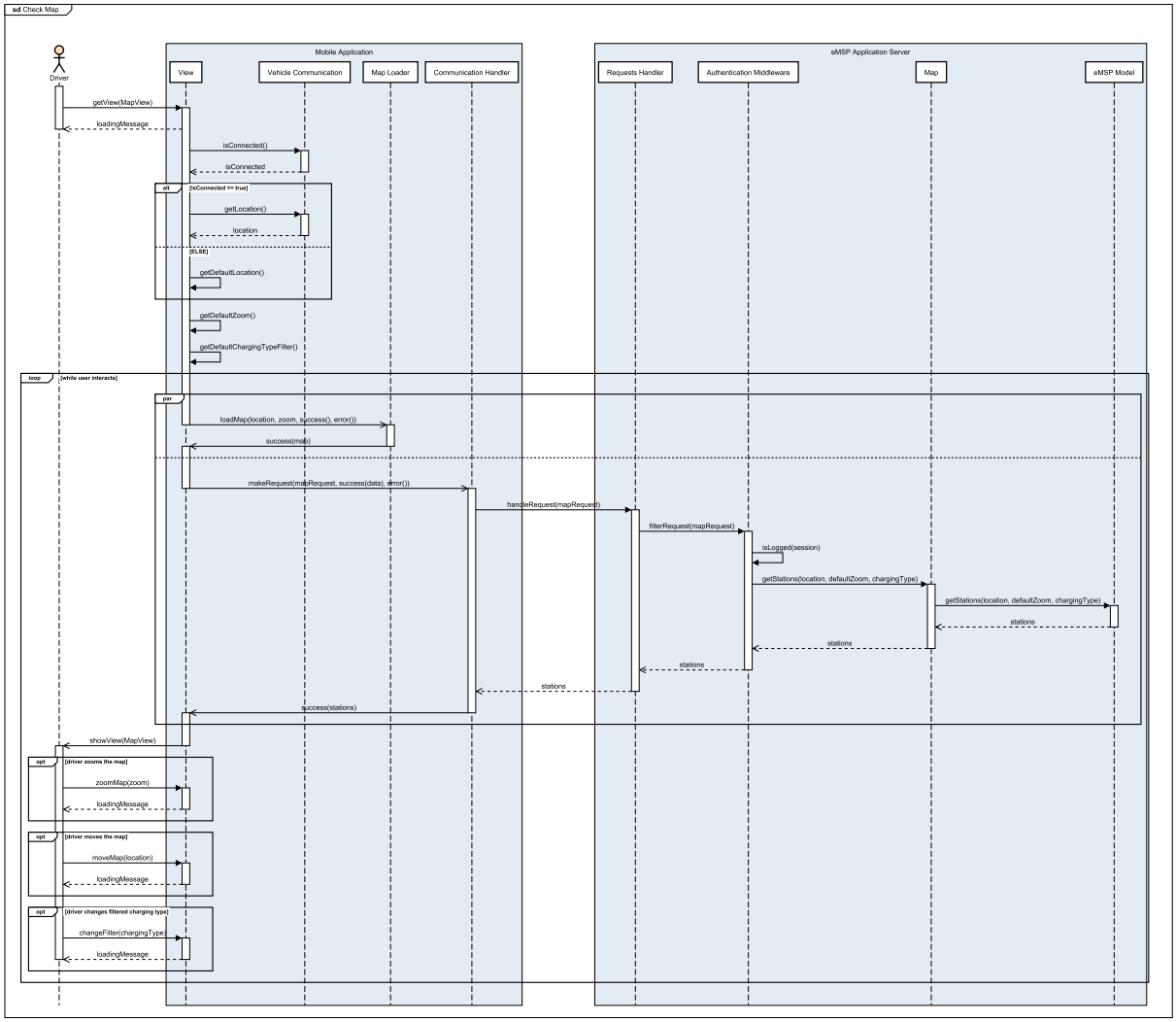

Figure 2.8: Driver interacts with map

The figure shows the process of the Driver checking the map and charging station present. If he is connected with his vehicle the application will show the charging station nearby his vehicle otherwise it will show a default location with default zoom and default charging type filter. (This can be decided during implementation). Every time one of the previous parameters is changed the app will ask the server for the data related to the parameters.

5. **Check Station Info**

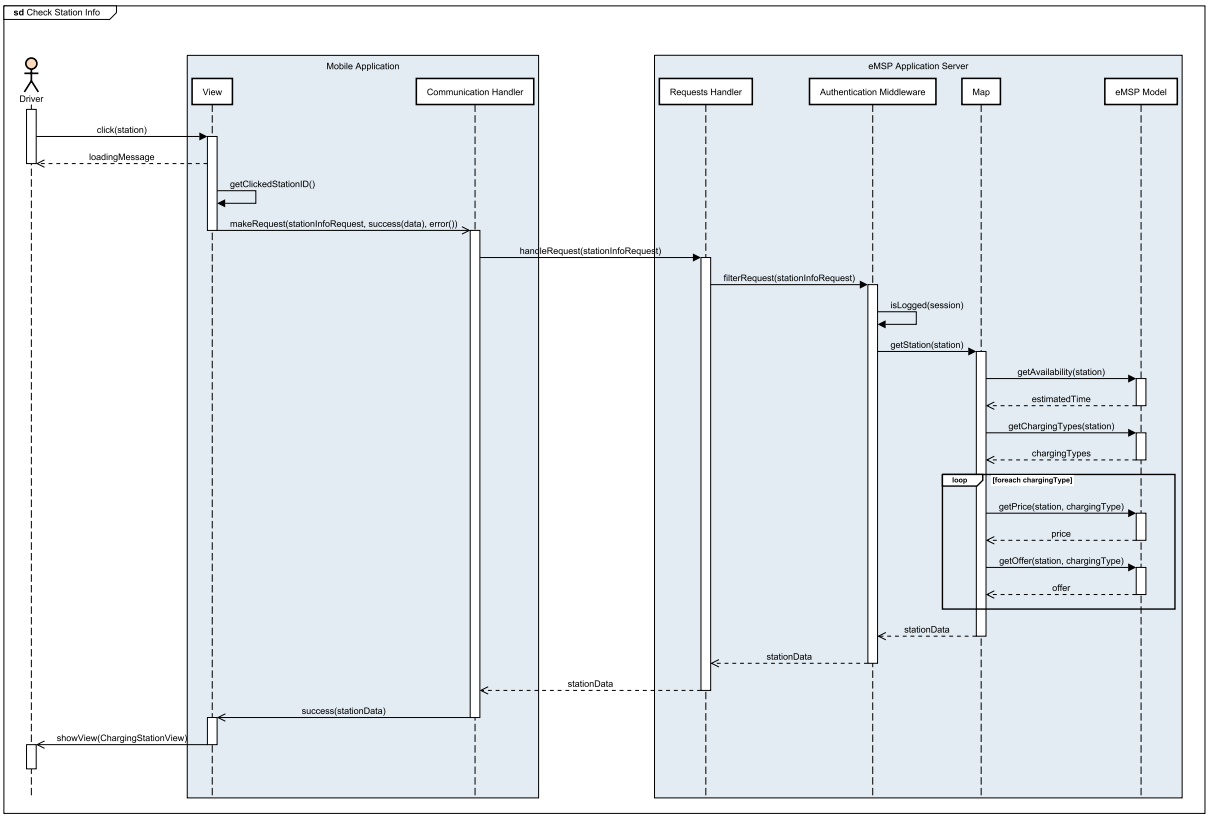

Figure 2.9: Driver checks a station's info

The figure shows the process of a Driver checking station info such as its availability, prices and offers.

6. **Create Booking**

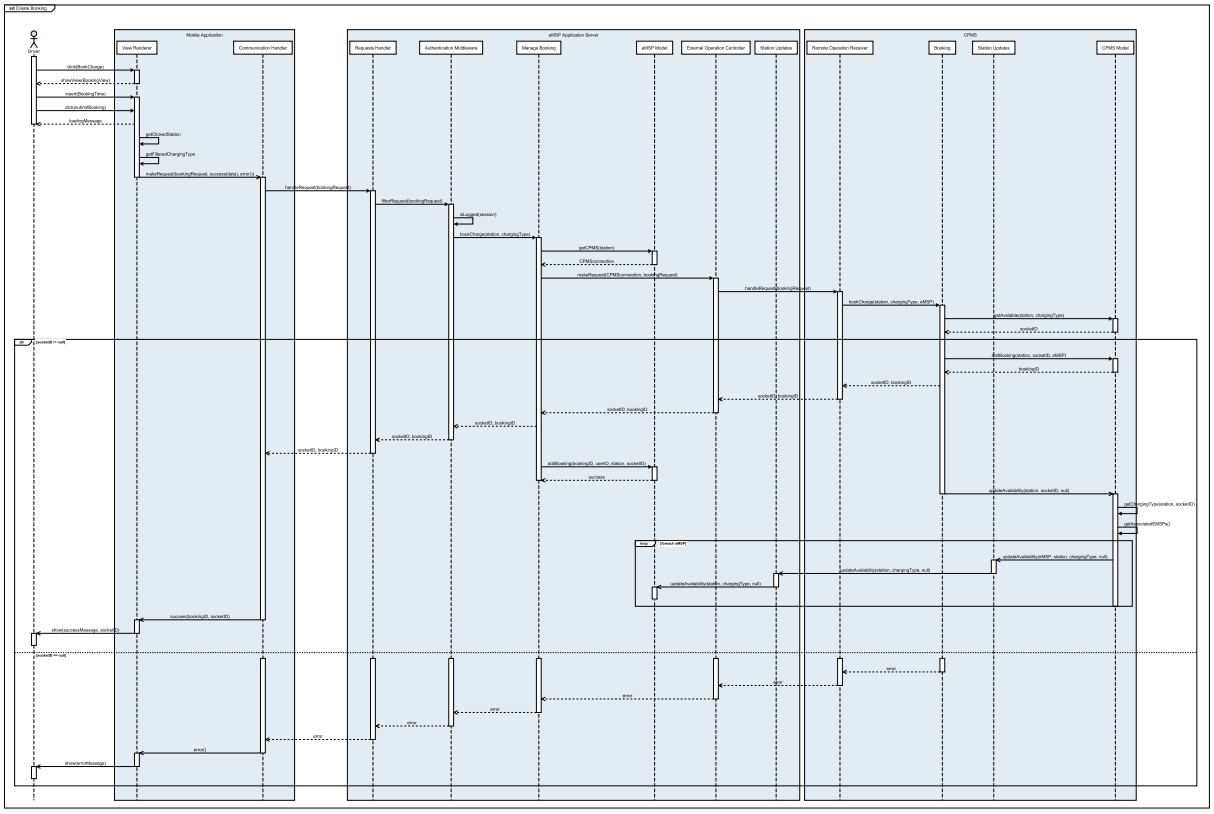

Figure 2.10: Driver creates a booking

The figure shows the process of the creation of a booking. After the Driver makes the request, the eMSP application server will process the request and make a new request to the specific CPMS that will answer (if there is no error) with the booked socket ID that will be shown to the Driver. Besides, since there is a change in station availability, CPMS updates all the associated eMSPs with the new data.

7. **Delete Booking**

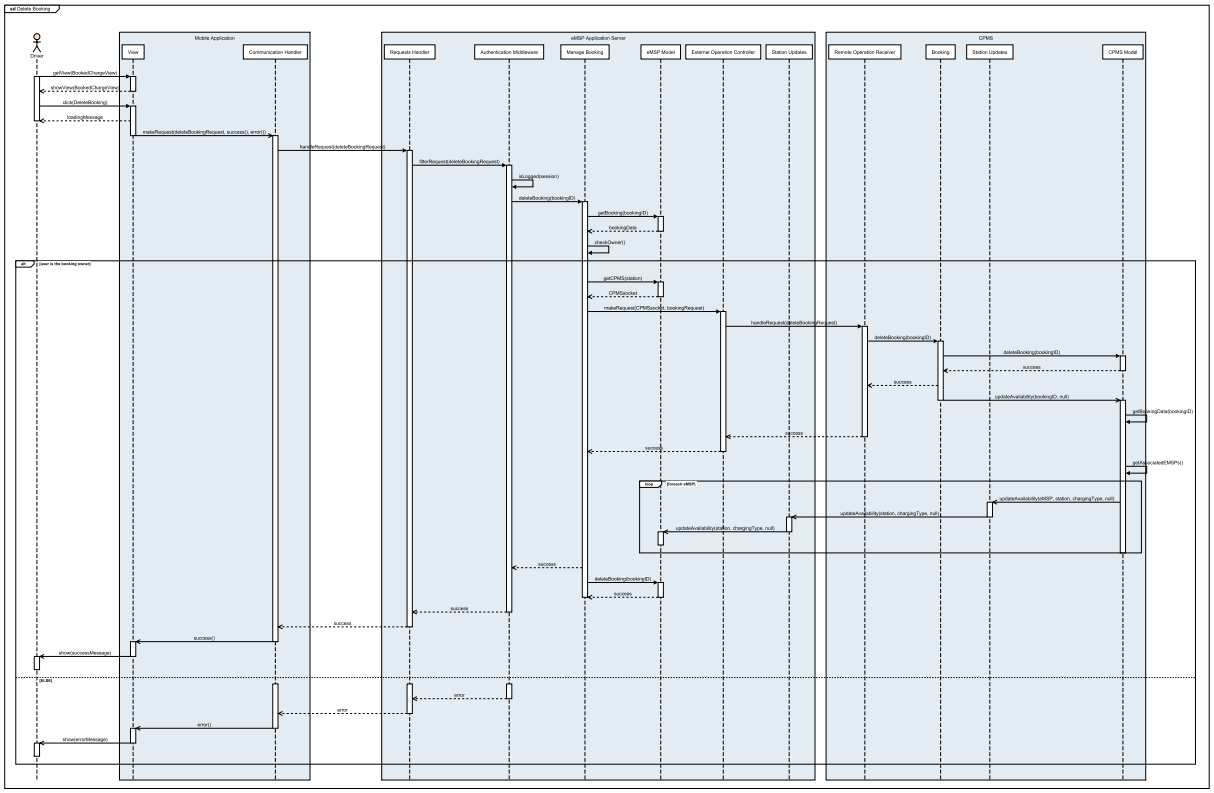

Figure 2.11: Driver deletes a booking

The figure shows the process of the deletion of a previously created booking. The process is similar to the previous one. When the CPMS updates the availability of a charging station, it sends a null estimated time value to indicate that the station is currently available.

8. **Suggestion Generation and Notification**

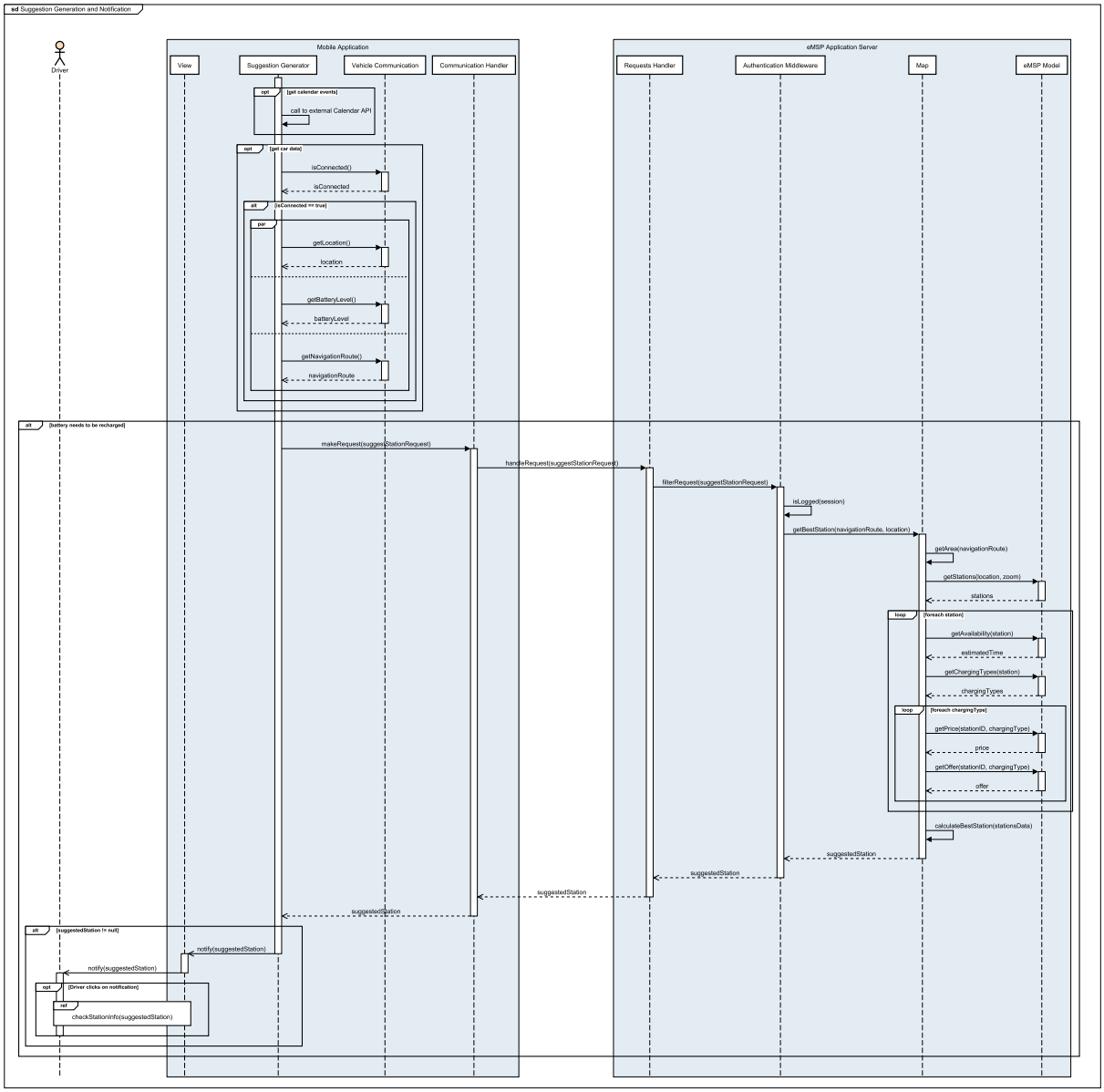

Figure 2.12: Driver receives and opens a suggestion

The figure shows the process for generating a suggestion. It will be generated on the mobile application by fetching data from the Driver's vehicle, Driver's calendar API and eMSP Application Server (in order to retrieve charging station info such as availability, prices and offers).

9. **Start Charging Process**

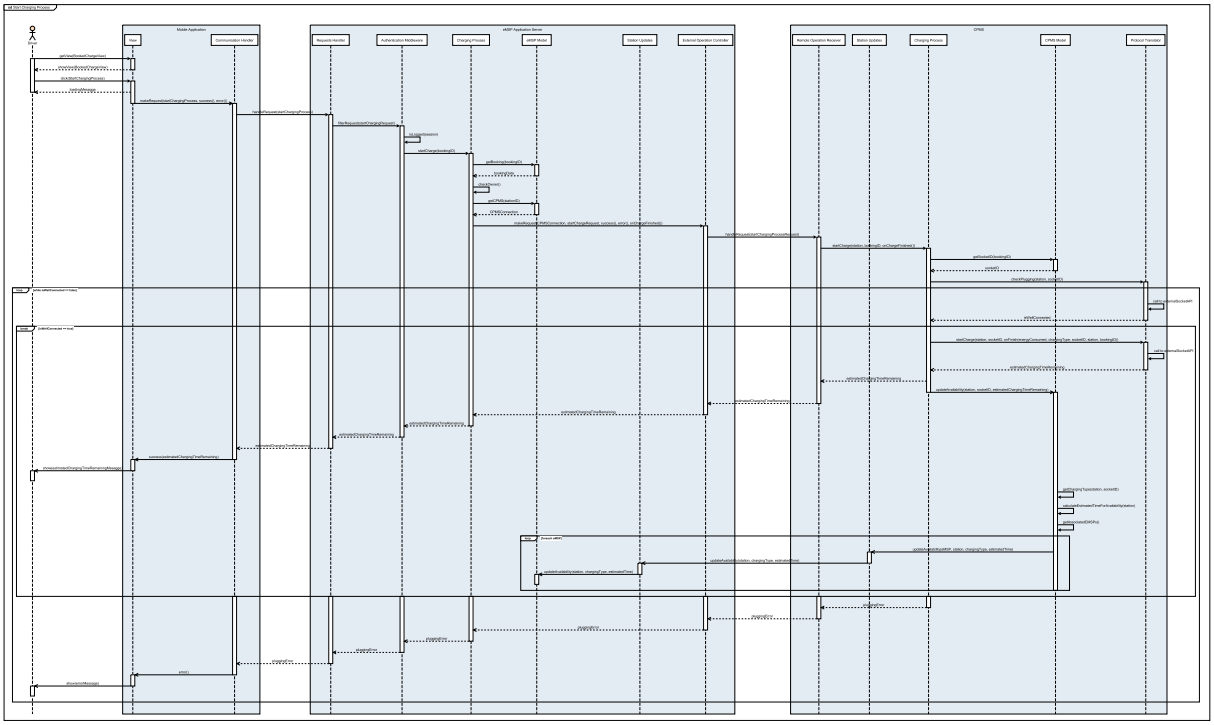

Figure 2.13: Driver starts the charging process

The figure shows the process for starting a charging process. After the Driver clicks the start charging button, the request will be sent to the eMSP Application Server and from there, to the specific CPMS that will make a check for correct plugging. The charging process won't start until the car is well connected. After it started, CPMS will answer the eMSP with the estimated time remaining, calculate the new station availability for the charging type used and update all the associated eMSPs.

10. **Stop Charging Process**

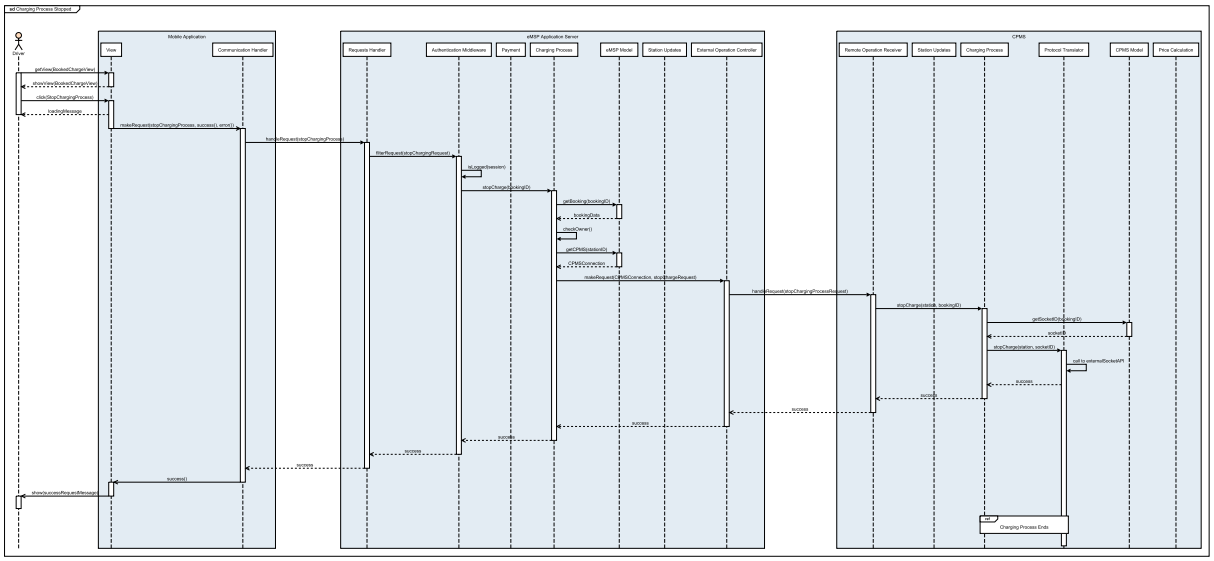

Figure 2.14: Driver stops the charging process

The figure shows the process for stopping a charging process. It starts when the Driver clicks on the stop charging button. After receiving the response, the Driver will receive as shown in the following sequence diagram the notification that the charging process has finished.

11. **Charging Process Ends**

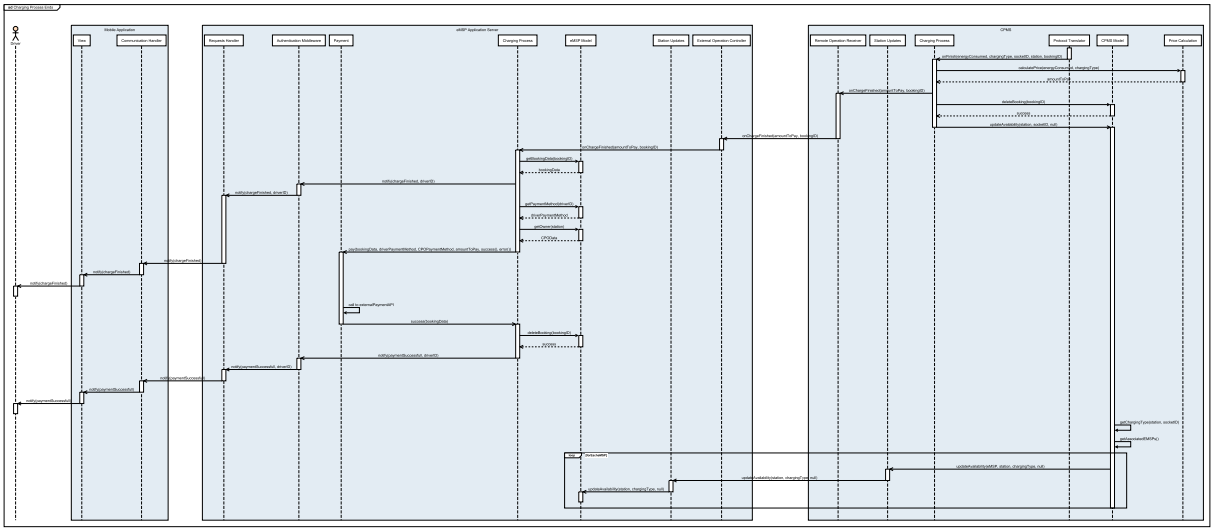

Figure 2.15: The charging process ends

The figure shows the process that is executed when a charging process ends. The *Protocol Translator* is going to be the component that will detect the end of a charging process and will start the corresponding flow. In particular, the CPMS will update every associated eMSPs for charging station availability and then, notify the specific eMSP that made the booking through an async request, with the amount needed to be paid. The eMSP will notify the Driver about the ending of the charging process and once completed the payment, it will send another notification to the Driver. This type of async call between eMSPs and CPMSs is advised to be implemented through some sort of callback function such as HTTP webhook. [7.2]

12. **CPO Authentication**

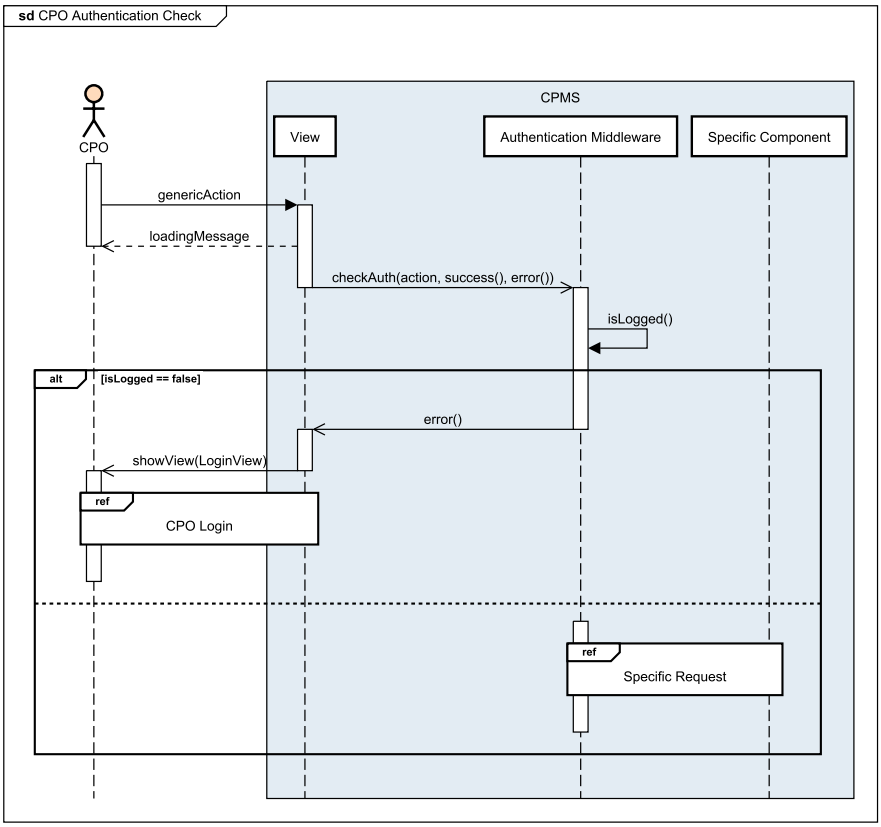

Figure 2.16: Authentication of a CPO

The figure shows the process that is executed for all the requests in order to authenticate CPO. The *isLogged()* function should be internal to the *Authentication Middleware* in order to avoid making calls to the DBSM or to the external eMall API. When the component can no longer check internally the validity, it will redirect the CPO to the Login View. More on this will be explained in the following sequence diagram.

13. **CPO Login**

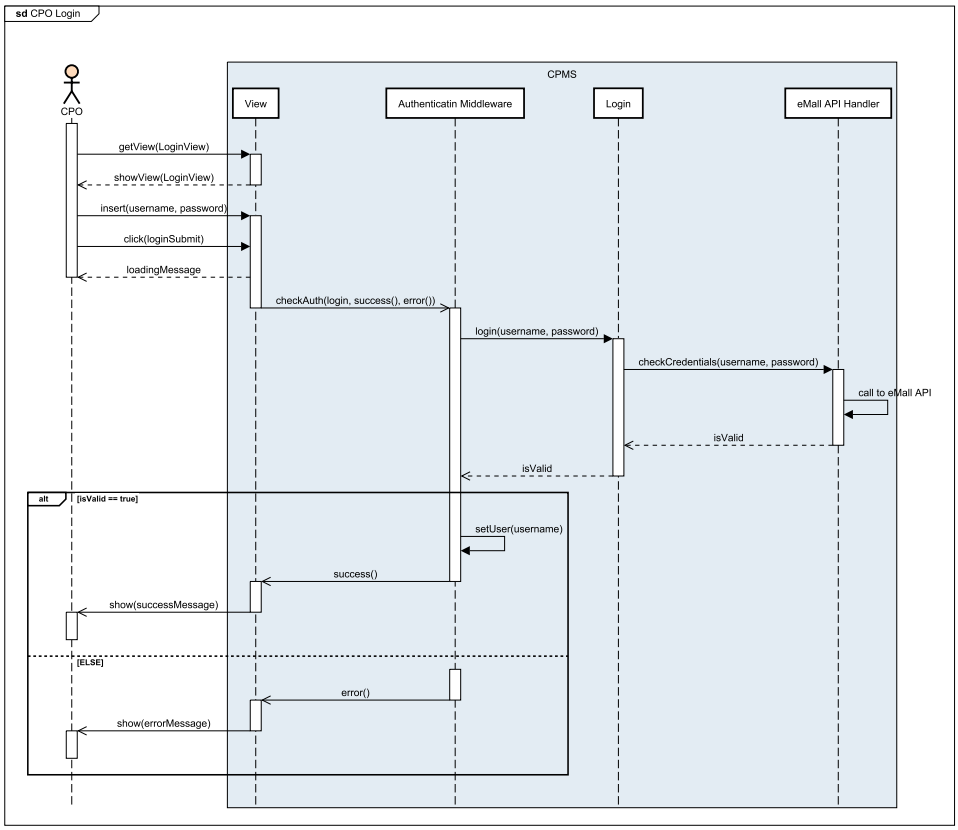

Figure 2.17: Login of a CPO

The figure shows the process for authenticating the CPO. Since the credentials are stored on the eMall platform, the CPMS should contact its API in order to perform validation. Once the validation is done, the *Authentication Middleware* should store internally that the CPO is authenticated and use that information when checking the authentication.

14. View Managed Station Status

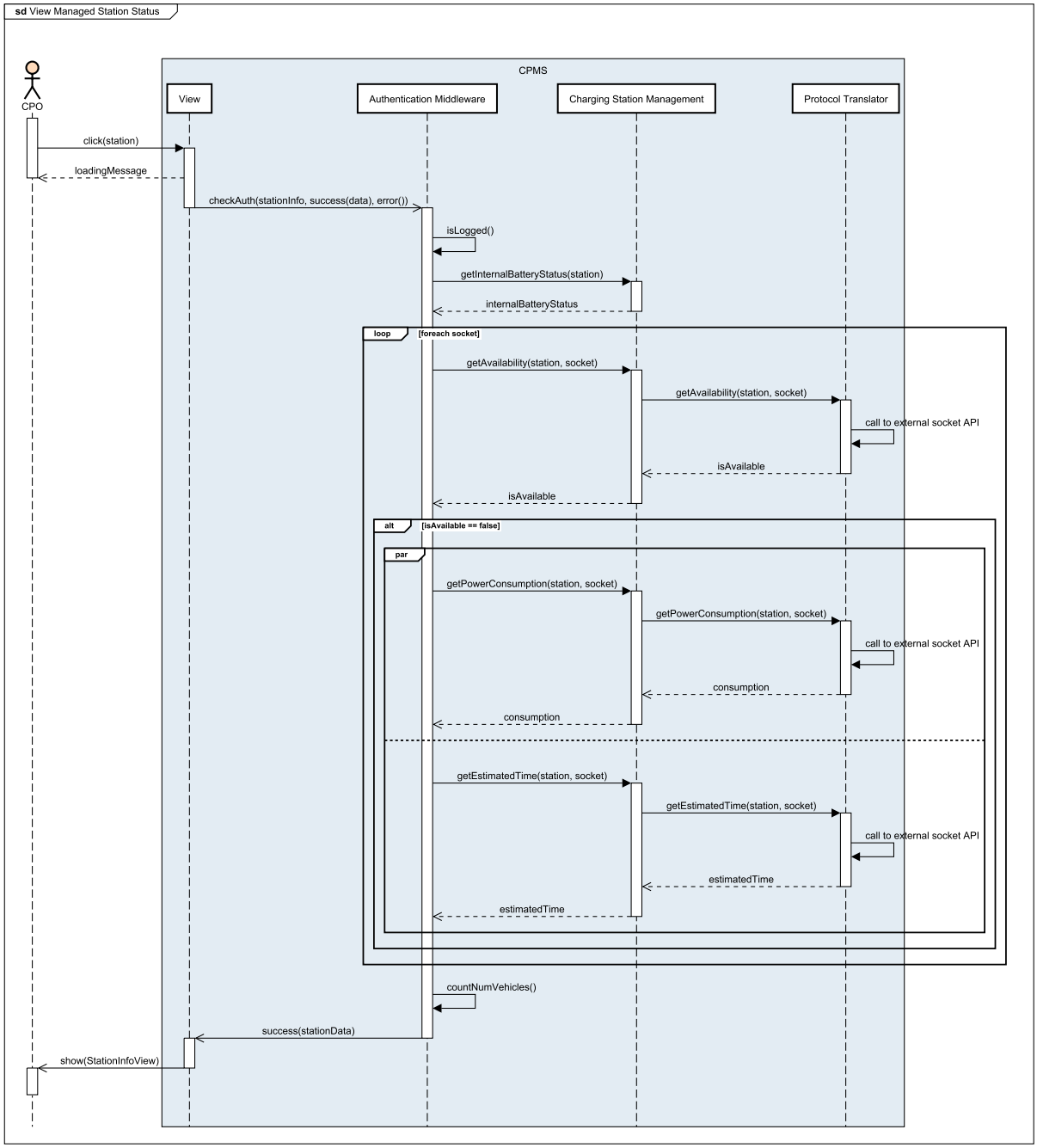

Figure 2.18: CPO views a managed station

The figure shows the process executed when the CPO wants to see the info of one of his stations managed by his CPMS such as sockets availability, internal battery status, the number of connected vehicles, their power consumption and estimated time to finish the charging process.

15. **Add Station To CPMS**

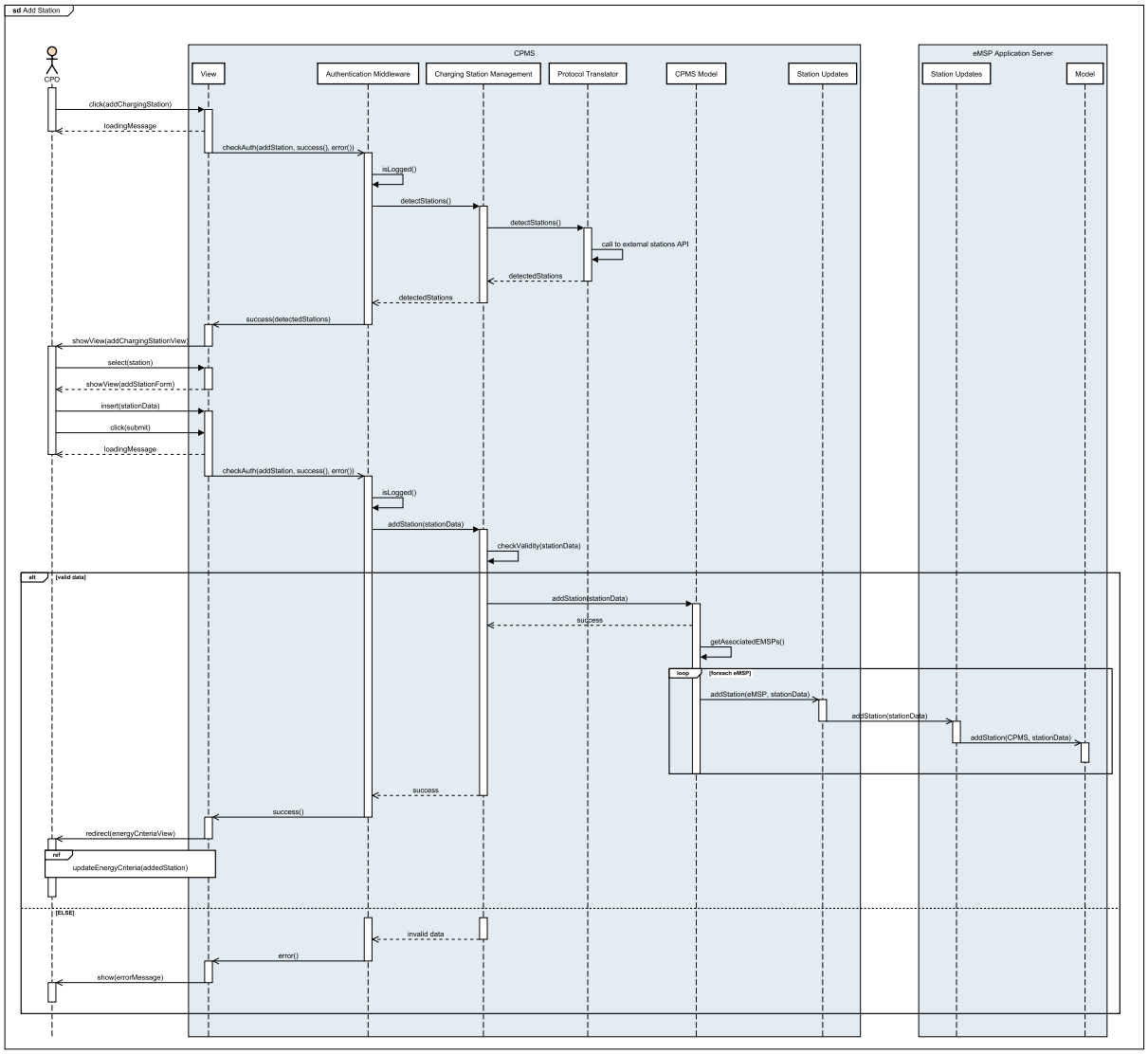

Figure 2.19: CPO adds a station

The figure shows the process for adding a new charging station. The CPMS, through its *Protocol Translator* component, will detect charging stations that can be added to the system and the CPO can add one of them by inserting all the required data. Once the station is added, the CPMS will update all the associated eMSPs and then redirect the CPO to the *energyCriteriaView*.

16. **Set Offer**

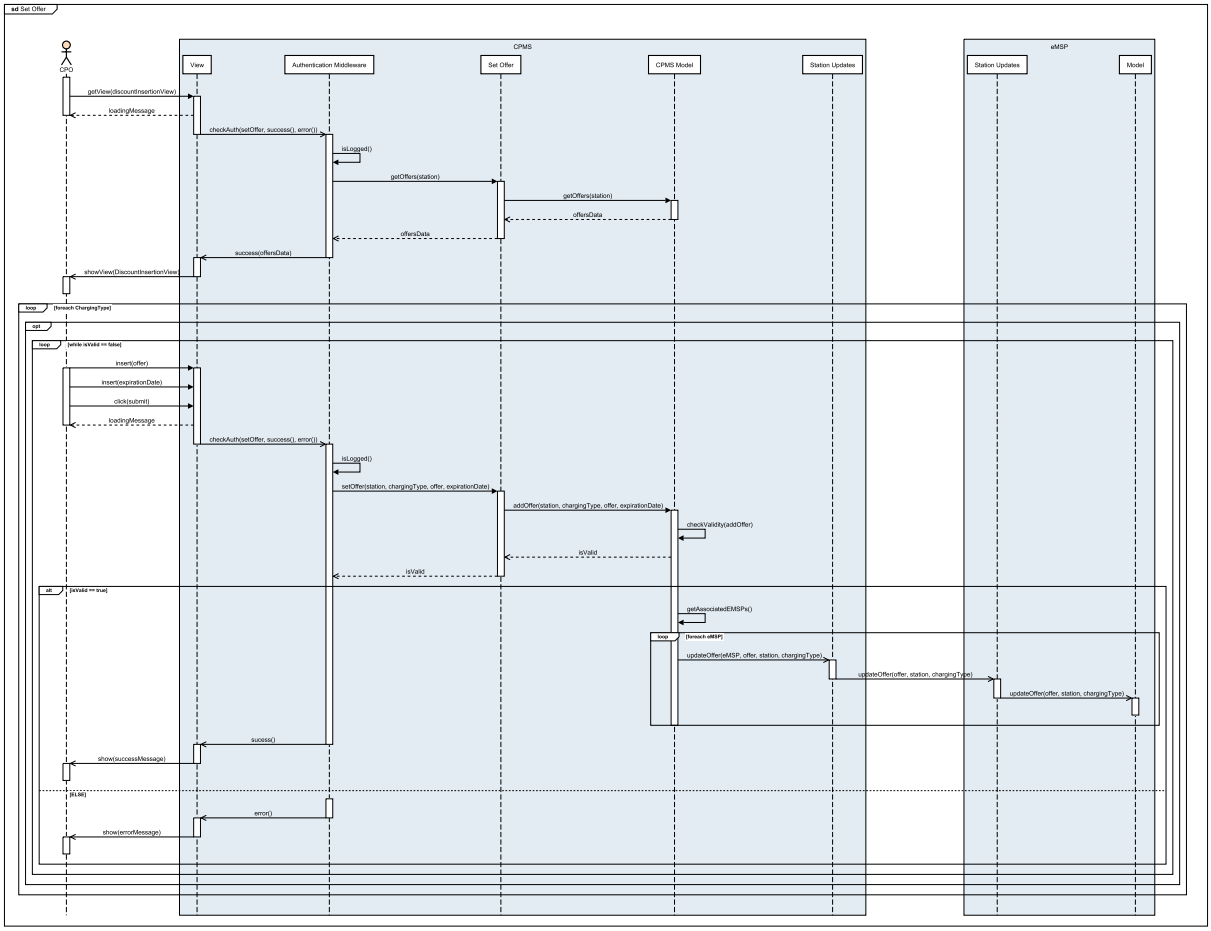

Figure 2.20: CPO sets a new offer

The figure shows the process when the CPO sets a new offer related to a charging station. After he sets the charging type, discount and expiration date for an offer, the CPMS will check the validity and will then update all the associated eMSPs

17. **Update Energy Criteria**

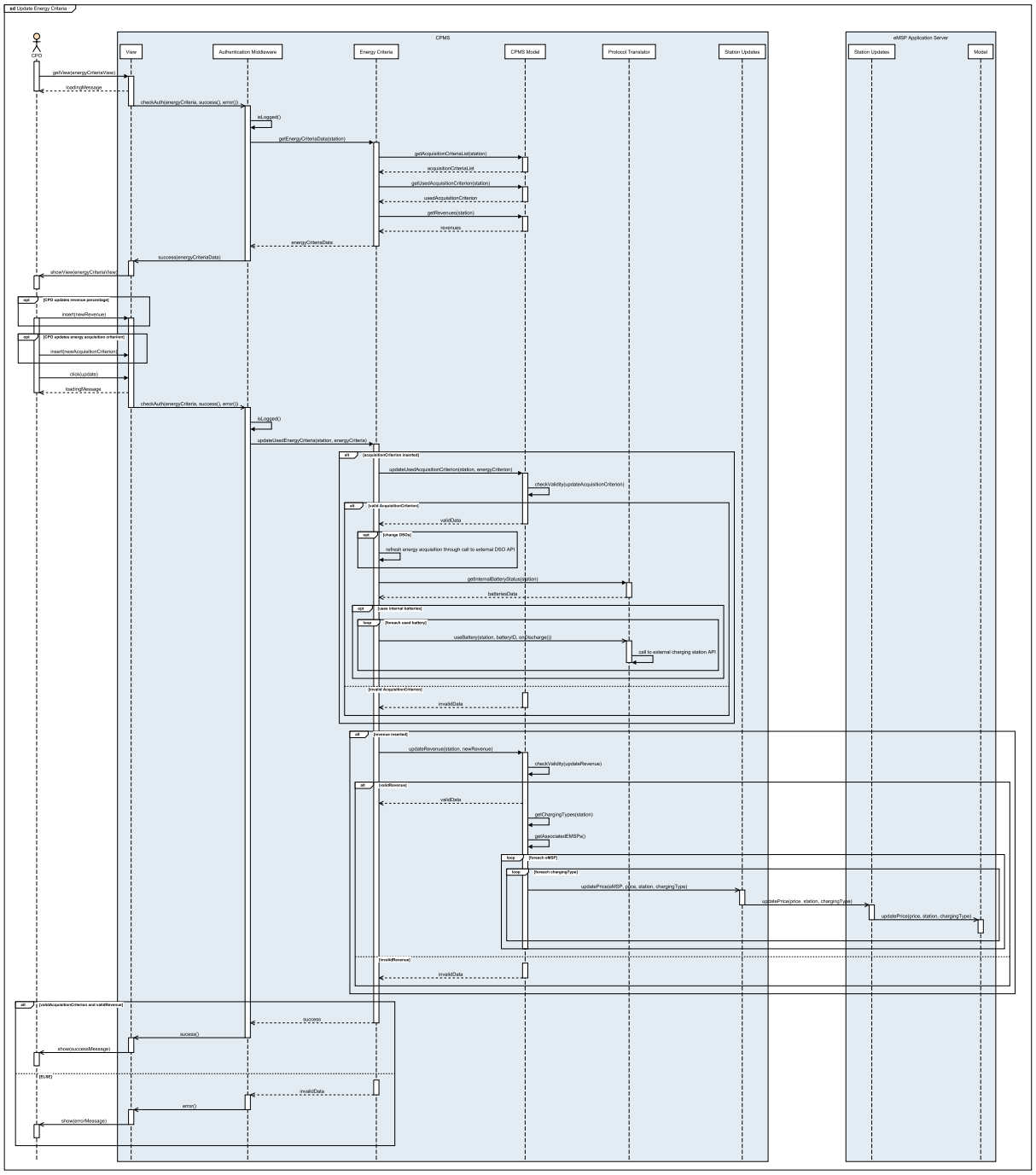

Figure 2.21: CPO updates energy criteria

The figure shows the process of the CPO updating the energy criteria (both acquisition and revenue) for a charging station. With the acquisition criterion, the CPO can select how the CPMS will automatically decide the best energy providers where to take energy from, respecting the chosen criterion, meanwhile, with the revenue, the CPO can set the percentage of gain that he will get from the services provided, based on the energy price.

18. **eMSP Association**

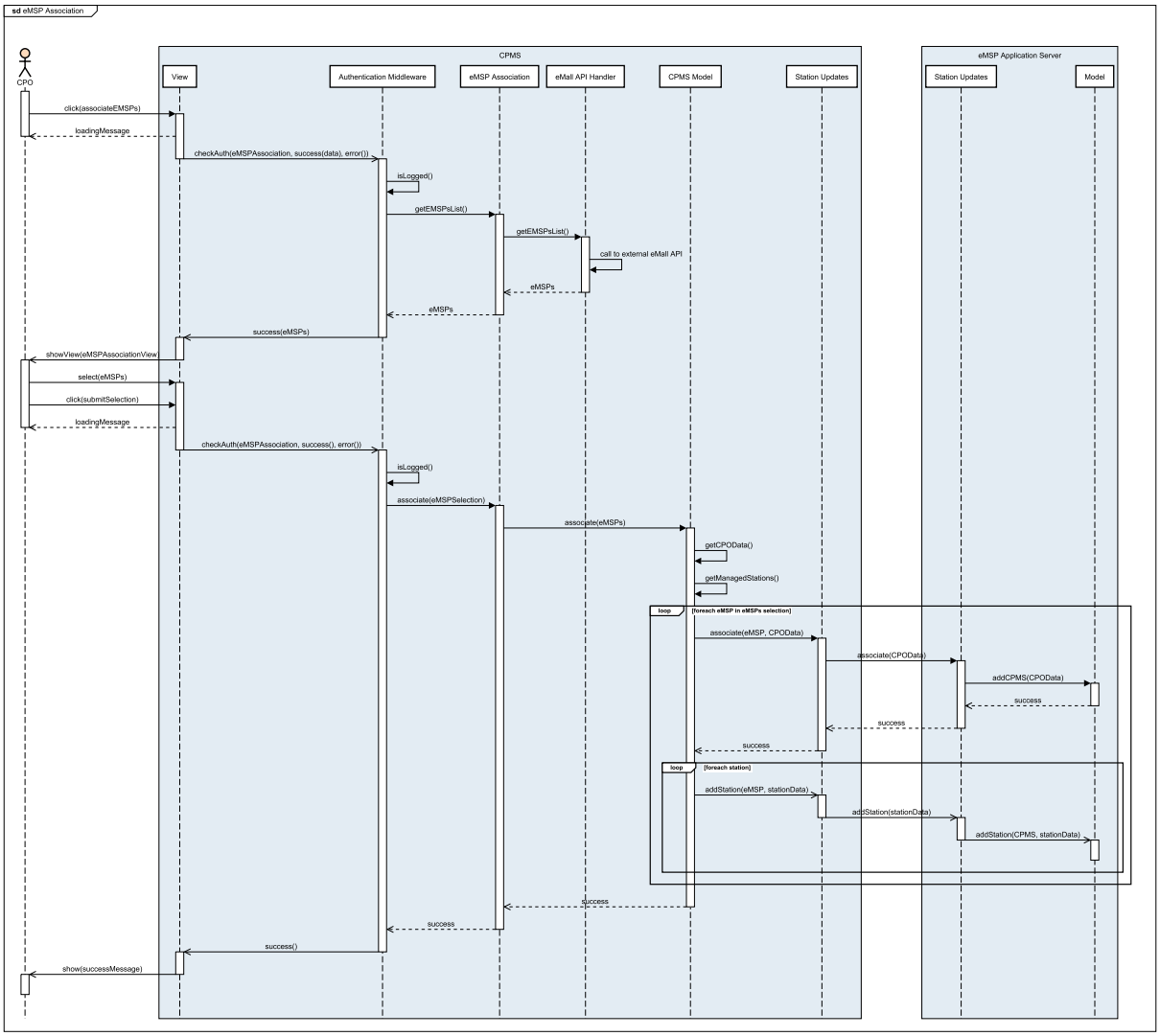

Figure 2.22: CPO associates new eMSPs

The figure shows the process of association between the CPMS and eMSPs. Without this process, the two subsystems won't be able to communicate. The first time a CPMS associates itself with an eMSP, it will also send all the existing data about its managed charging stations.

2.5. Component Interfaces

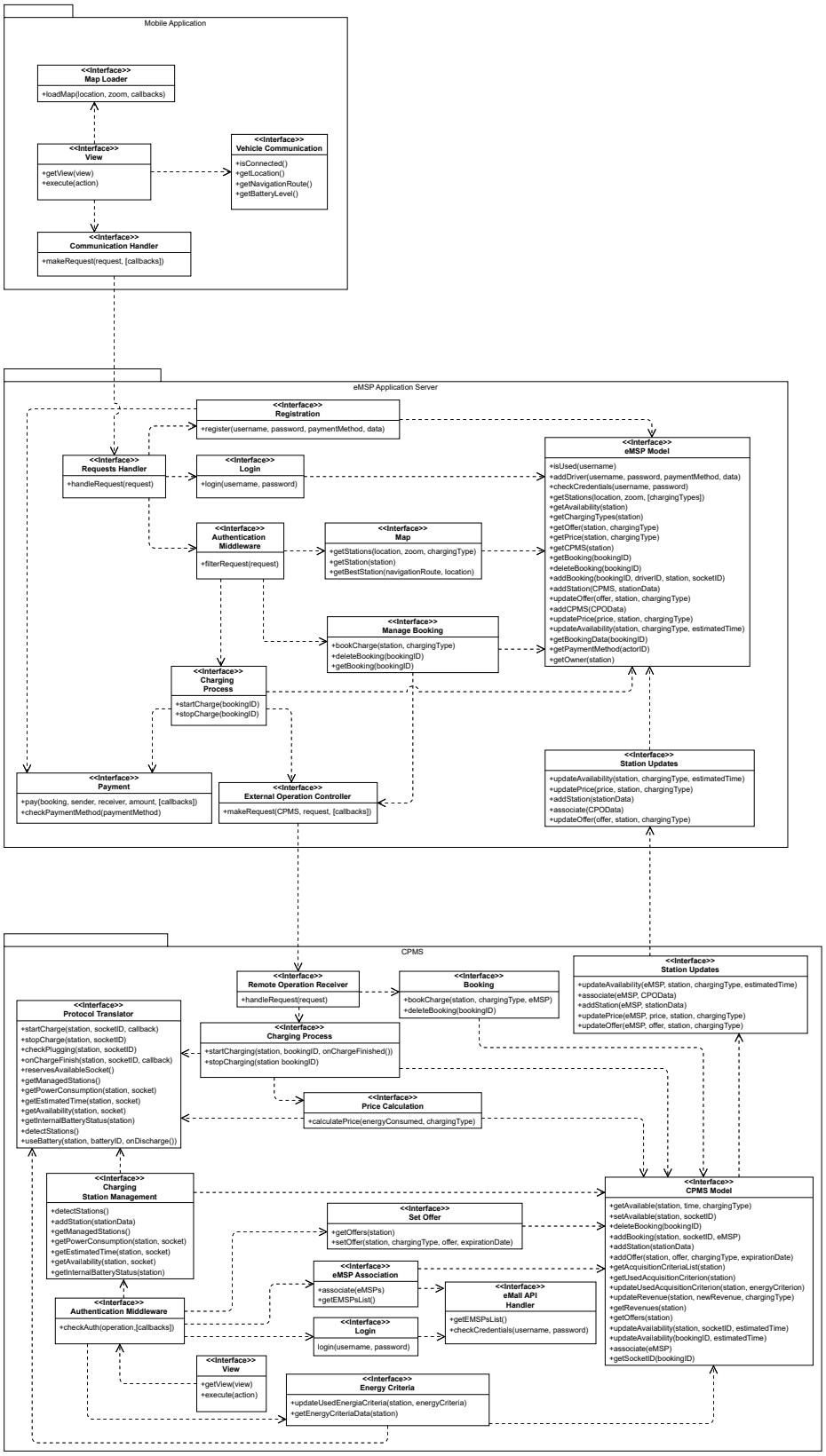

Figure 2.23: Component Interfaces Diagram

2.6. Architectural Styles and Patterns

2.6.1. Architectural Styles

In the overview section, a description is provided of how a three-tier client-server architecture has been used to design the eMSP system, while a monolithic architecture has been used to design the CPMS.
For communication between the eMSP Mobile Application, eMSP server, and CPMS, a REST architecture was chosen. In particular, the communication can be integrated using a webhook [7.2] to allow for HTTP callbacks.

2.6.2. Patterns

Suggested patterns to implement the application:

- **Model-View-Controller Pattern (MVC) :** It is an architectural pattern which divides the application into three interconnected parts. It is used to separate how the information is represented inside the system from what the user actually sees. It is a really good and easy pattern to use when a system is based on a three-tier architecture.
- **Observer Pattern (OP)**: It simplifies the communications between objects, notifying changes in the state of other specific objects. For example, it could be adopted by the Station Update component of the CPMS to observe the model and then notify any changes regarding charging stations data to all associated eMSPs.
- **Model-View-ViewModel (MVVM):** This pattern is similar to MVC, but the ViewModel component is responsible for exposing the data and logic of the model in a way that is consumable by the view. This can make it easier to develop and test the user interface for the mobile application.

2.7. Other Design Decisions

This system is designed on the constraint that it will interact with external services. From these external services, the system needs to fetch data or perform operations. In particular, the following are the interfaces that both the eMSP and CPMS are going to use:

Mobile Application

- **Map Provider API:** From this interface, the mobile application can fetch the data needed to show the Driver the map where charging stations will be viewed on.
- **Calendar API:** From this interface, the mobile application can fetch the Driver's calendar events needed to create more personalized suggestions.
- **Vehicle Connection:** From this interface, the mobile application can fetch data such as navigation routes, the battery level and the location needed to create more personalized suggestions for the Driver and for positioning the eMSP map where the Driver is.

eMSP

- **Payment API:** From this interface, the eMSP can handle all the payments between Drivers and CPOs without needing to implement a payment system from scratch.
- **DBMS API:** From this interface, the eMSP can perform operations in order to store data and read them. An idea would be to use a framework like JPA (Java Persistence API) [7.2] to keep the database synced with the model component.

CPMS

- **External Charging Station/Socket API:** From this interface, the CPMS can communicate with charging stations or sockets in order to perform operations or receive notifications about a specific event happening.
- **eMall API:** From this interface, the CPMS can fetch the data of all the eMSPs in order to associate itself with them and also check the authenticity of a CPO.
- **DSO API:** From this interface, the CPMS can fetch all the information provided by a DSO about the energy. It is important to remind that all the DSOs have homogenous APIs
- **DBMS API:** From this interface, the CPMS can perform operations in order to store data and read them. An idea would be to use a framework like JPA (Java Persistence API) [7.2] to keep the database synced with the model component.

3 | USER INTERFACE DESIGN

3.1. eMSP Application

Figure 3.1: Main Page

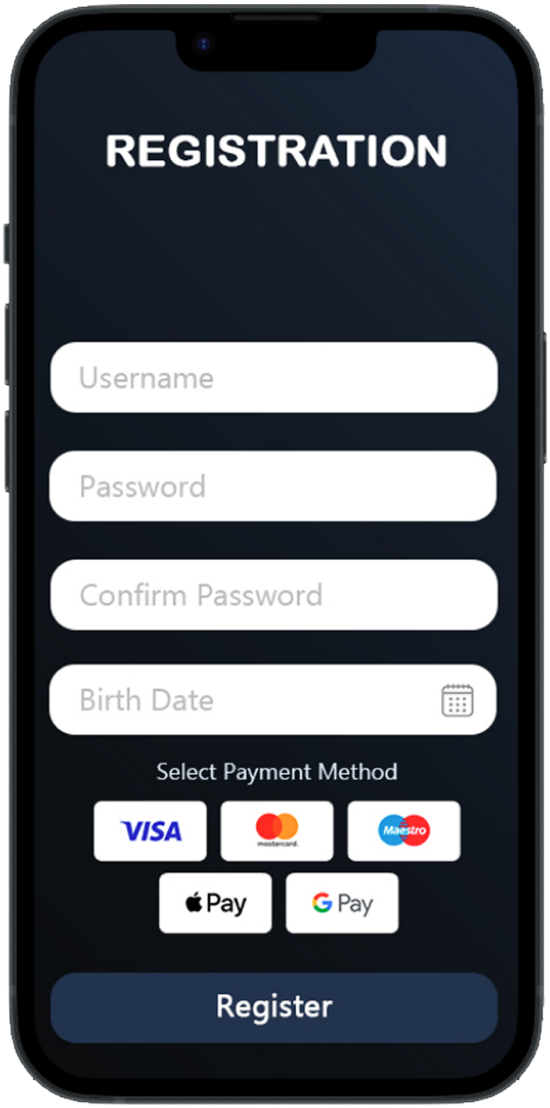

Figure 3.2: Registration Page

These two mockups display the main page on the left, which has two buttons that redirect to the login and registration pages on the right. The registration page allows Unregistered Drivers to sign up for the eMSP by submitting their personal data and a payment method.

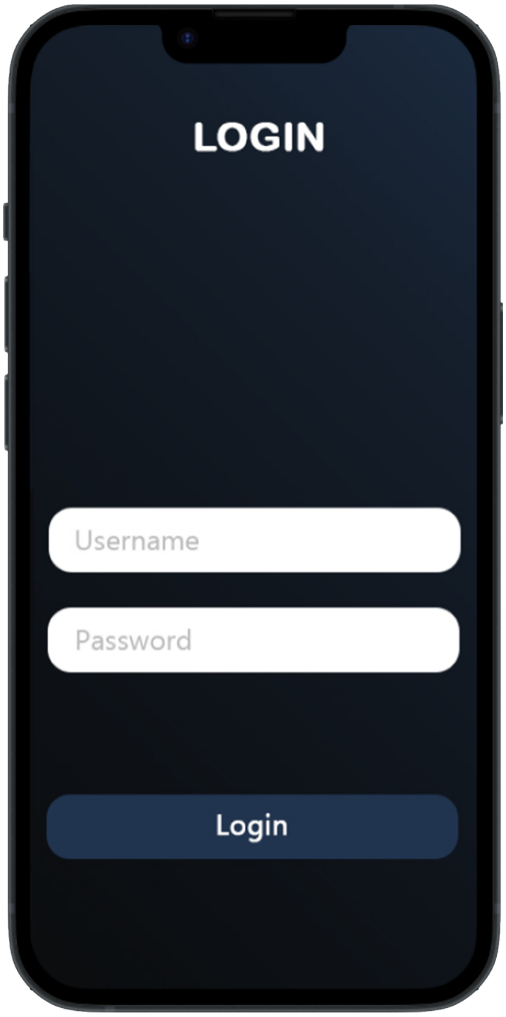

Figure 3.3: Login Page

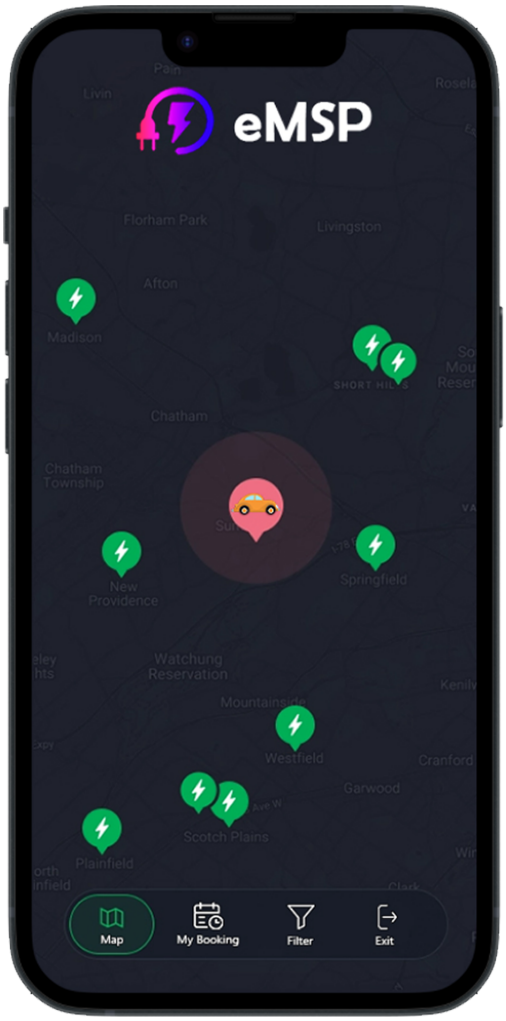

Figure 3.4: Map Page

The image on the left depicts the login page for the Driver, while the image on the right shows the map page of the eMSP mobile application. This page allows the Driver to navigate the map, set filters, click on stations, and access his booking page.

Figure 3.5: Charging Station Info Page

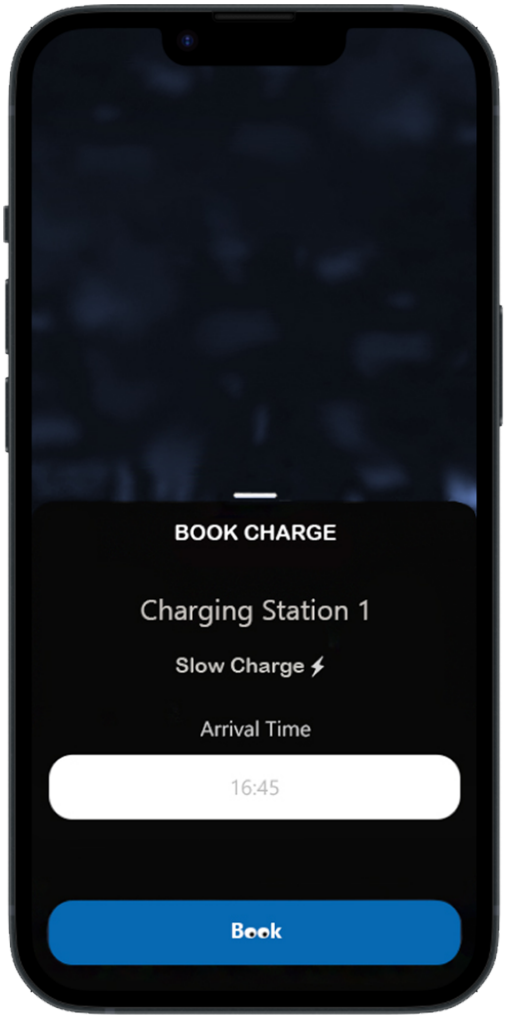

Figure 3.6: Book Charge Page

These two mockups depict the charging station page on the left and the station's book charge page on the right. The charging station page allows the Driver to view information about the station, while the book charge page enables the Driver to book a charge for the selected charging type at the chosen arrival time.

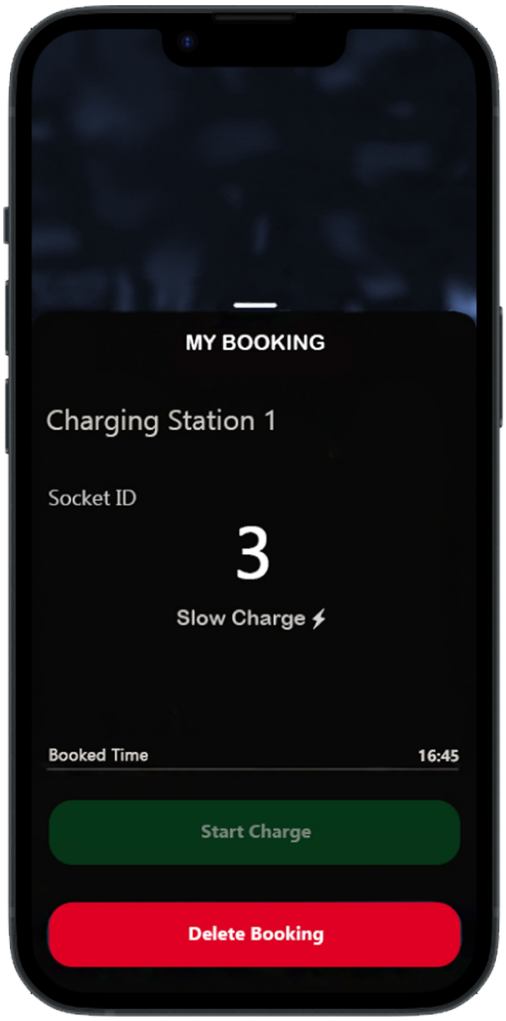

Figure 3.7: My Booking Page

Figure 3.8: Charging Process Page

The mockups depict the charging process functionality. On the left, the personal booking page (accessible from the map page) is shown, where the Driver can view their booked charge details and delete it before the arrival time. After the arrival time, the start charge button becomes active and the Driver can initiate the charging process. On the right image, the charging view is depicted, with the estimated time to complete the charge and the option to interrupt it prematurely by pressing the 'Stop Charge' button.

3.2. CPMS

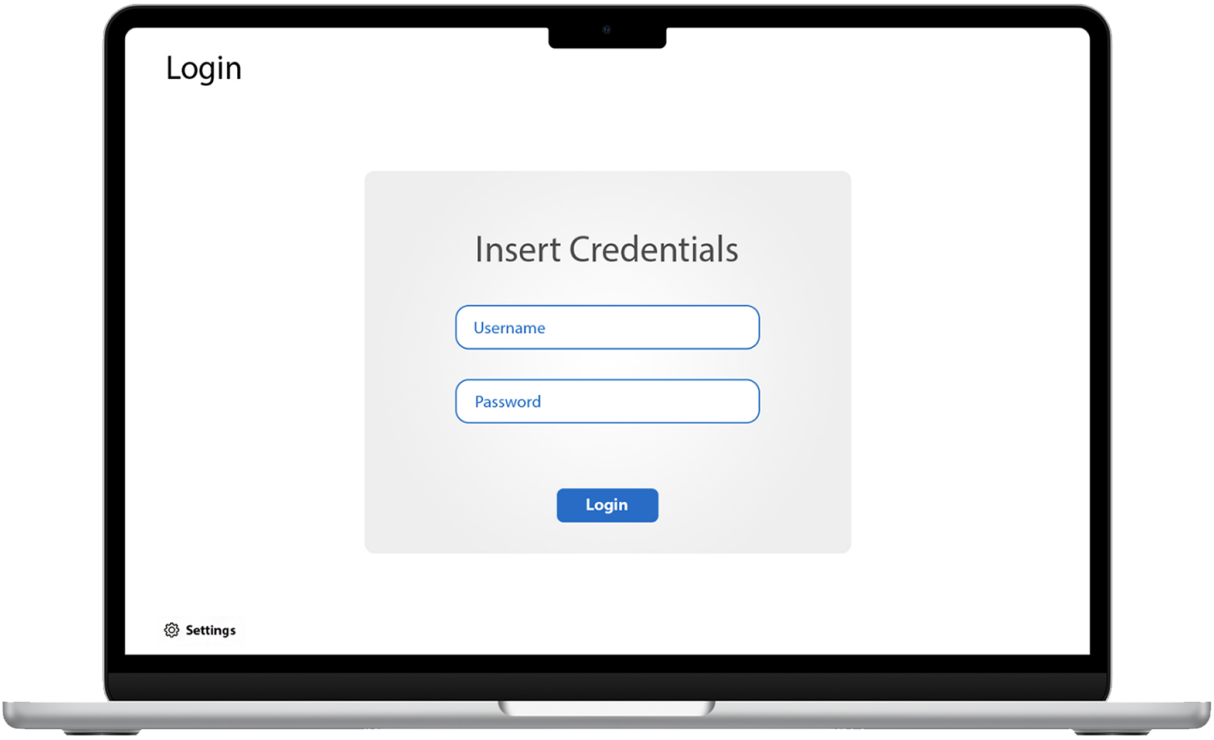

Figure 3.9: Login Page

The mockup shows the login page where the CPO can access the CPMS by submitting his credentials.

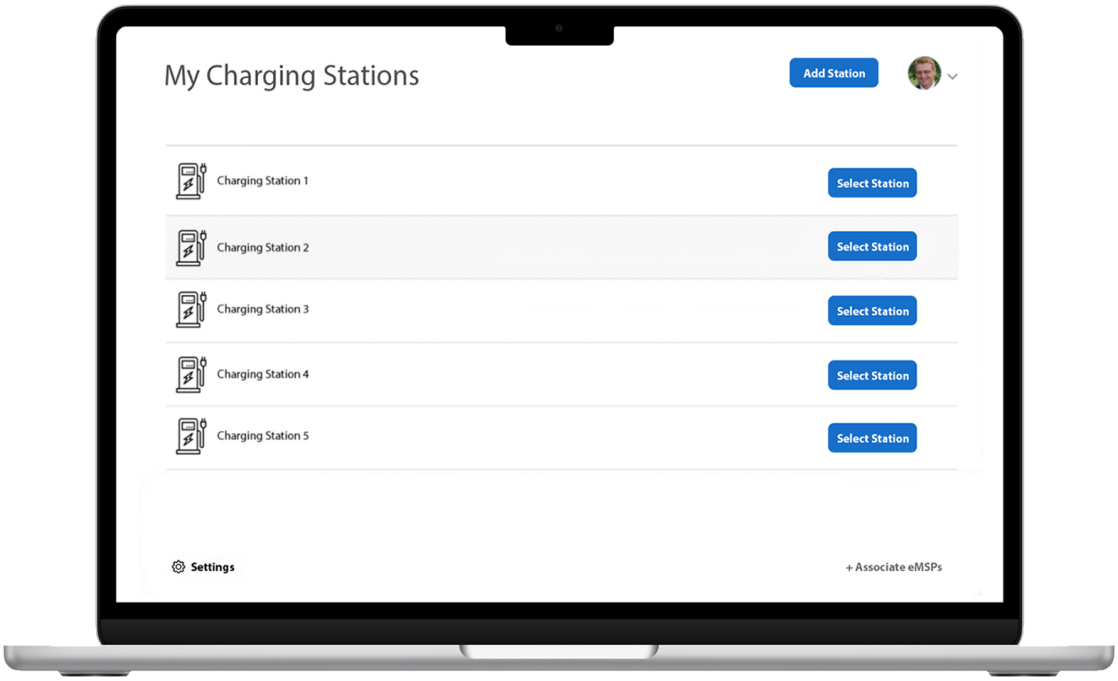

Figure 3.10: Home Page

The mockup shows the home page of the CPMS, from which the CPO can select his managed stations to monitor them, add a new station or associate new eMSPs.

Figure 3.11: Charging Station Info Page

The mockup shows an info page for a charging station, accessible from the main page. Here the CPO can see the status of the station's sockets and batteries. Also, he can update the station's energy criteria and set a new offer.

Figure 3.12: Update Criteria Page

The mockup shows the energy criteria page, accessible from the station info page, from which the CPO can update the energy acquisition criteria and the revenue for the selected charging station.

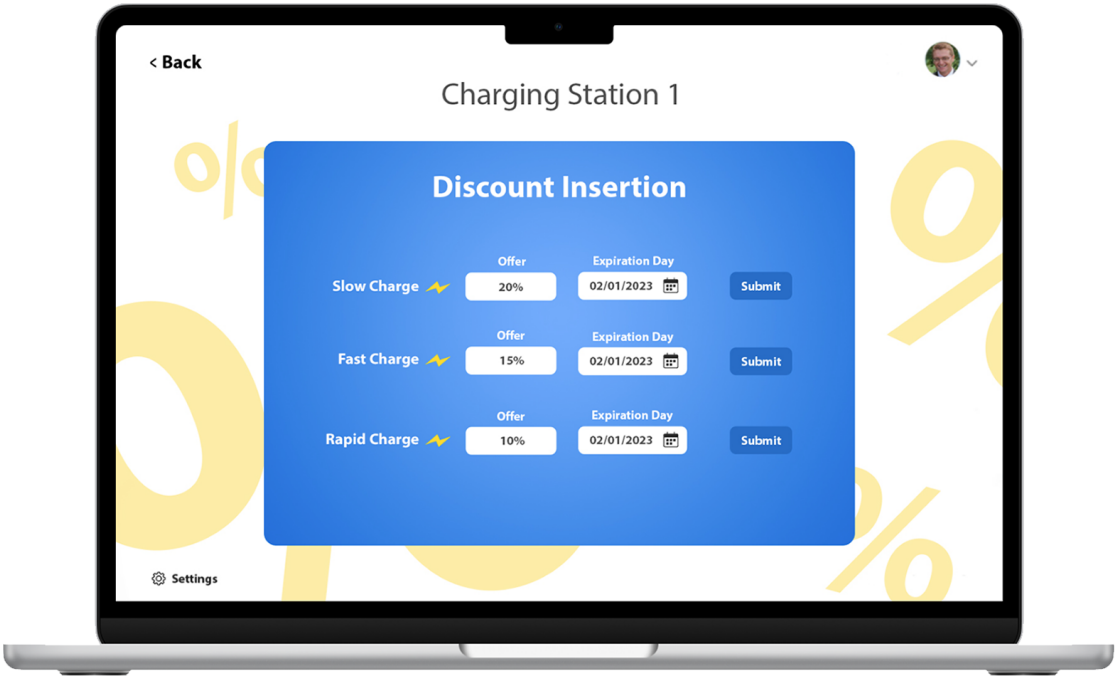

Figure 3.13: Set Offer Page

The mockup shows the set offer page, accessible from the station info page, from which the CPO can set an offer for each individual charging type supported by that charging station and also insert its expiration date.

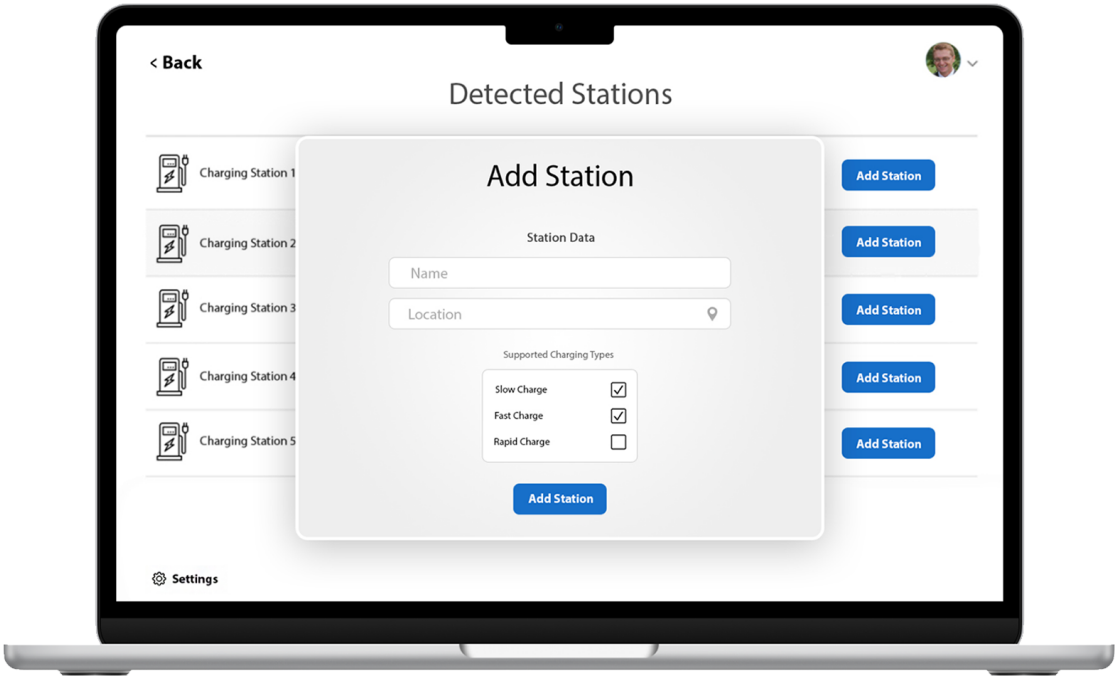

Figure 3.14: Add Station Page

The mockup shows the add station page, accessible from the main page, from which the CPO can add a new detected charging station to the ones managed by the CPMS, inserting its name, location and supported charging types.

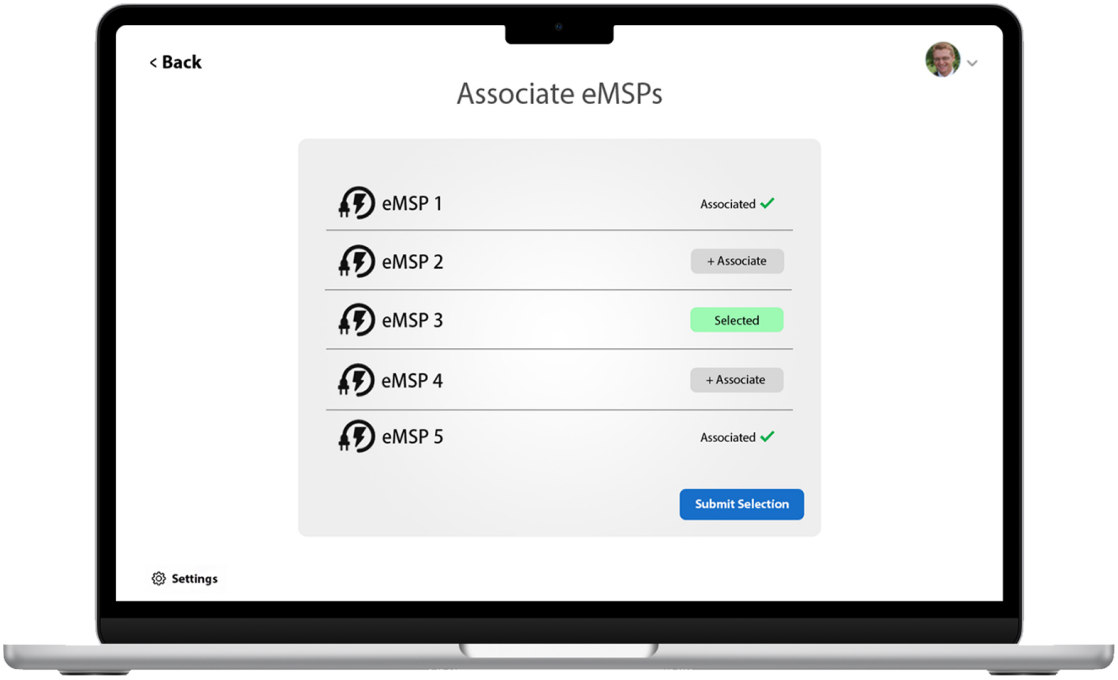

Figure 3.15: Associate eMSPs Page

The mockup shows the eMSP association page, accessible from the main page, from which the CPO can select the eMSPs with which he wants its CPMS to associate. On the same page, he can also see the already associated ones.

4 | REQUIREMENTS TRACEABILITY

4.1. Explicit Mapping On Goals

G1	**Allow Drivers to check nearby charging stations and see info about their prices, special offers and availability.**
FRE1	An Unregistered Driver shall be able to register himself on the eMSP application.
FRE2	The Driver shall be able to login to the eMSP with his credentials.
FRE3	The system shall notify the Driver if there is no internet connection.
FRE4	The Driver shall be able to see the location of all the charging stations in a certain geographic area.
FRE5	The Driver shall be able to select the charging types he is interested in while searching for a charging station in order to filter them.
FRE6	The Driver shall be able to check the cost per kWh for the specified charging types in a specific charging station.
FRE7	The Driver shall be able to see which charging stations are currently available for the specified charging types.
FRE8	The Driver shall be able to see an estimated time in which a specific charging station will become available.
FRE18	Every time the system fails to elaborate an operation related to a Driver, then he shall receive a notification containing the error details.

FRE22	If the vehicle is connected, the system shall be able to retrieve the Driver's vehicle location
FRE45	The system shall allow the CPO to apply discounts for the energy prices of a managed charging station until a set expiration date.
FRE46	The CPO shall be able to see a list of all the existing eMSPs.
FRE47	The CPO shall be able to decide which eMSP he wants to associate with his CPMS in order to permit communication between the two systems
FRE55	If CPMS can calculate the estimated time before each charging type becomes available, CPMS updates the availability of a charging station with the estimated time otherwise, CPMS updates only the availability.
FRE56	CPMS updates the price of a charging type in a station.
FRE57	CPMS updates the offer of a charging type in a station
FRE58	CPMS update about the data of the newly created charging station such as location, prices, and supported charging types.
	Mobile Application • Map Loader • View • Communication Handler • Vehicle Communication

	eMSP • Requests Handler • Login • Registration • Authentication Middleware • Communication Handler • Map • Station Updates • eMSP Model
	CPMS • Station Updates • CPMS Model • View • Authentication Middleware • eMall API Handler • eMSP Association • Station Updates • CPMS Model

G2	**Allow Drivers to create and delete bookings for charging their vehicle in a charging station.**
FRE1	An Unregistered Driver shall be able to register himself on the eMSP application.
FRE2	The Driver shall be able to login to the eMSP with his credentials.
FRE3	The system shall notify the Driver if there is no internet connection.
FRE4	The Driver shall be able to see the location of all the charging stations in a certain geographic area.
FRE5	The Driver shall be able to select the charging types he is interested in while searching for a charging station in order to filter them.

FRE7	The Driver shall be able to see which charging stations are currently available for the specified charging types.
FRE8	The Driver shall be able to see an estimated time in which a specific charging station will become available.
FRE9	The Driver shall be able to book a charging socket in an available charging station, from 0 up to a max specified amount of time before.
FRE10	The system shall show the Driver the booked socket ID number.
FRE11	The Driver shall be able to delete a booked charging process before its starting time.
FRE12	The Driver shall receive a notification that confirms the booking went well.
FRE18	Every time the system fails to elaborate an operation related to a Driver, then he shall receive a notification containing the error details.
FRE46	The CPO shall be able to see a list of all the existing eMSPs.
FRE47	The CPO shall be able to decide which eMSP he wants to associate with his CPMS in order to permit communication between the two systems
FRE48	The eMSP shall be able to book a charge and receive the booked charging socket on a single CPMS
FRE49	The eMSP shall be able to delete a previously booked charge on a single CPMS
FRE55	If CPMS can calculate the estimated time before each charging type becomes available, CPMS updates the availability of a charging station with the estimated time otherwise, CPMS updates only the availability.
FRE58	CPMS updates about the data of the newly created charging station such as location, prices and supported charging type.

Mobile Application • Map Loader • View • Communication Handler
eMSP • Requests Handler • Login • Registration • Communication Handler • Authentication Middleware • Map • Manage Booking • External Operation Controller • Station Updates • eMSP Model
CPMS • Station Updates • View • Authentication Middleware • eMall API Handler • eMSP Association • Station Updates • Remote Operation Receiver • Booking • CPMS Model

G3	**Allow Drivers to manage and monitor their charging process.**
FRE1	An Unregistered Driver shall be able to register himself on the eMSP application.
FRE2	The Driver shall be able to login to the eMSP with his credentials.
FRE3	The system shall notify the Driver if there is no internet connection.
FRE10	The system shall show the Driver the booked socket ID number
FRE13	As soon as the booked time starts the Driver shall be able to start the recharging process.
FRE14	During the recharging process the Driver shall be able to stop it in advance.
FRE15	The Driver shall be able to check the estimated remaining time until the end of the charging process.
FRE16	The system shall notify the Driver when the charging process is complete.
FRE18	Every time the system fails to elaborate an operation related to a Driver, then he shall receive a notification containing the error details.
FRE36	The system shall be able to give an estimation about how much time is needed to end the charging process for a particular vehicle being charged in a charging station.
FRE46	The CPO shall be able to see a list of all the existing eMSPs.
FRE47	The CPO shall be able to decide which eMSP he wants to associate with his CPMS in order to permit communication between the two systems
FRE50	The eMSP shall be able to start a charge on a single CPMS.
FRE51	The eMSP shall be notified when the vehicle is not well connected to the charging socket on a single CPMS
FRE52	The eMSP shall be able to stop a charge on a single CPMS.

FRE53	The eMSP shall be notified with the required amount of money to pay when the charging process ends on a single CPMS .
FRE54	The eMSP shall be able to ask for the estimated ending time for a specific charging process.
	Mobile Application • View • Communication Handler
	eMSP • Requests Handler • Login • Registration • Communication Handler • Authentication Middleware • Map • Booking • Charging Process • External Operation Controller • Station Updates • eMSP Model

	CPMS • Station Updates • View • Authentication Middleware • eMall API Handler • eMSP Association • Station Updates • Remote Operation Receiver • Booking • Charging Process • Payment Calculation • CPMS Model

G4	**Allow Drivers to pay CPOs for the charging process provided.**
FRE1	An Unregistered Driver shall be able to register himself on the eMSP application.
FRE2	The Driver shall be able to login to the eMSP with his credentials.
FRE3	The system shall notify the Driver if there is no internet connection.
FRE17	At the end of the charging process, the system shall handle the payment between the Driver and the CPO through an external API.
FRE18	Every time the system fails to elaborate an operation related to a Driver, then he shall receive a notification containing the error details.
FRE19	The system shall notify the Driver when a successful payment occurs.
FRE40	The CPO shall be able to select the revenue percentage amount from the energy sale of each charging station.

FRE42	The system shall be able, for each charging station, to automatically calculate the selling price of the energy based on the revenue criteria and on the price of the energy provider.
FRE46	The CPO shall be able to see a list of all the existing eMSPs.
FRE47	The CPO shall be able to decide which eMSP he wants to associate with his CPMS in order to permit communication between the two systems
FRE53	The eMSP shall be notified with the required amount of money to pay when the charging process ends on a single CPMS .
	Mobile Application • View • Communication Handler
	eMSP • Requests Handler • Login • Registration • Charging Process • Payment • Station Updates • eMSP Model

	CPMS • View • Authentication Middleware • eMall API Handler • eMSP Association • Station Updates • Charging Process • Payment Calculation • CPMS Model

G5	**Proactively suggest to the Drivers where to go to charge their vehicle.**
FRE18	Every time the system fails to elaborate an operation related to a Driver, then he shall receive a notification containing the error details.
FRE20	The system shall be able to check if there exists a connection between the Driver's vehicle and the system itself.
FRE21	If the vehicle is connected, the system shall be able to retrieve the battery level of the Driver's vehicle.
FRE22	If the vehicle is connected, the system shall be able to retrieve the Driver's vehicle location.
FRE23	The Driver shall be able to authorize the system to access his personal data such as his calendar, navigation system and location.
FRE24	The system shall be able to retrieve data from the Driver's personal calendar.
FRE25	The system shall retrieve information about the Driver's navigation system.

FRE26	If the vehicle is connected to the Driver's device, the eMSP shall create charging suggestion notifications based on the status of the vehicle's battery, the Driver's schedule (his calendar and navigation system), special offers made available at some charging stations and the availability of charging sockets at those identified charging stations.
FRE27	The system shall advise the Driver with a notification when and where he should recharge his vehicle's battery.
FRE28	The Driver shall be able to click on the charging suggestion notification sent by the eMSP and that shall redirect to the suggested charging station's info page.
FRE45	The system shall allow the CPO to apply discounts for the energy prices of a managed charging station until a set expiration date.
FRE46	The CPO shall be able to see a list of all the existing eMSPs.
FRE47	The CPO shall be able to decide which eMSP he wants to associate with his CPMS in order to permit communication between the two systems
FRE55	If CPMS can calculate the estimated time before each charging type becomes available, CPMS updates the availability of a charging station with the estimated time, otherwise CPMS updates only the availability.
FRE56	CPMS updates the price of a charging type in a station.
FRE57	CPMS updates the offer of a charging type in a station.
FRE58	CPMS update about the data of the newly created charging station such as location, prices, supported charging types.
	Mobile Application • View • Communication Handler • Vehicle Communication • Suggestion Generator

	eMSP • Requests Handler • Authentication Middleware • Map • eMSP Model
	CPMS • View • Authentication Middleware • eMall API Handler • eMSP Association • Station Updates • Set Offer • CPMS Model

G6	**Allow CPOs to manage the charging prices and the criteria of energy acquisition in their charging stations.**
FRE29	The CPO shall be able to login to the CPMS with his credentials.
FRE30	The system shall notify the CPO if there is no internet connection.
FRE31	The CPO shall be able to see the list of only all his owned charging stations.
FRE32	The CPO shall be able to add a new charging station.
FRE37	The system shall be able to retrieve the current energy price and the capacity for each of the available DSOs.
FRE38	The system shall be able to use the batteries as energy providers with prices equal to the cost that was used to recharge them.
FRE39	The CPO shall be able to select the criteria that will be used by the CPMS in order to acquire energy, such as Energy provider with the lowest price, Energy Provider with biggest energy capacity.

FRE40	The CPO shall be able to select the revenue percentage amount from the energy sale of each charging station.
FRE41	The system shall be able, for each charging station, to automatically decide from which energy provider to retrieve the energy from, based on the criteria chosen by the CPO.
FRE42	The system shall be able, for each charging station, to automatically calculate the selling price of the energy based on the revenue criteria and on the price of the energy provider.
FRE44	The system shall be able to automatically decide whether to store or not energy in the internal batteries, if available, of one of the managed charging stations.
FRE45	The system shall allow the CPO to apply discounts for the energy prices of a managed charging station until a set expiration date.
	CPMS • View • Authentication Middleware • Login • eMall API Handler • Charing Station Management • Energy Criteria • Set Offer • CPMS Model

G7	**Allow CPOs to connect to all their charging stations and monitor them.**
FRE29	The CPO shall be able to login to the CPMS with his credentials.
FRE30	The system shall notify the CPO if there is no internet connection.
FRE31	The CPO shall be able to see the list of only all his owned charging stations.
FRE32	The CPO shall be able to add a new charging station.
FRE33	The CPO shall be able to check the remaining energy stored in the batteries in one of his own charging stations.

FRE34	The CPO shall be able to check the number of vehicles being charged in one of his charging stations.
FRE35	The CPO shall be able to check the amount of power absorbed from each vehicle being charged in one of his charging stations.
FRE36	The system shall be able to give an estimation about how much time is needed to end the charging process for a particular vehicle being charged in a charging station.
FRE50	The eMSP shall be able to start a charge on a single CPMS.
FRE52	The eMSP shall be able to stop a charge on a single CPMS.
FRE54	The eMSP shall be able to ask for the estimated ending time for a specific charging process.
	eMSP • Charging Process • External Operation Controller • eMSP Model
	CPMS • View • Authentication Middleware • Login • eMall API Handler • Charing Station Management • Protocol Translator • Charging Process • Remote Operation Receiver • CPMS Model

4.2. Mapping On Requirements

In the following sub sections, the mapping between a specific requirement and components is shown. In order to have a more concise table, the name of components are shorted

4.2.1. Mobile Application

Acronyms

- **V** View
- **VC** Vehicle Communication
- **ML** Map Loader
- **CH** Comunication Handler
- **SG** Suggestion Generator

Table

ID	V	VC	ML	CH	SG
FRE1	X			X	
FRE2	X			X	
FRE3	X				
FRE4	X		X	X	
FRE5	X		X	X	
FRE6	X			X	
FRE7	X		X	X	
FRE8	X			X	
FRE9	X			X	
FRE10	X			X	
FRE11	X			X	
FRE12	X			X	
FRE13	X			X	
FRE14	X			X	
FRE15	X			X	
FRE16	X			X	
FRE18	X			X	
FRE19	X			X	
FRE20		X			

FRE21		X			
FRE22		X			
FRE23	X				
FRE24					X
FRE25		X			
FRE26		X			X
FRE27	X				X
FRE28	X				

4.2.2. eMSP Application Server

Acronyms

- **RH** Request Handler
- **LO** Login
- **RE** Registration
- **AM** Authentication Middleware
- **MA** Map
- **MB** Manage Booking
- **CP** Charging Process
- **PA** Payment
- **EO** External Operation Controller
- **SU** Station Updates
- **MO** eMSP Model

Table

ID	RH	LO	RE	AM	MA	MB	CP	PA	EO	SU	MO
FRE1	X		X								X
FRE2		X	X								X
FRE4	X			X	X						X
FRE5	X			X	X						X
FRE6	X			X	X						X
FRE7	X			X	X						X
FRE8	X			X	X						X

FRE9	X			X		X					X
FRE10	X			X		X					X
FRE11	X			X		X					X
FRE12	X			X		X					X
FRE13	X			X			X				X
FRE14	X			X			X				X
FRE15	X			X			X				X
FRE16							X				X
FRE17							X	X			X
FRE19	X							X			X
FRE26	X			X	X						X
FRE47										X	X
FRE48						X			X		X
FRE49						X			X		X
FRE50							X		X		X
FRE51							X		X		X
FRE52							X		X		X
FRE53							X		X		X
FRE54							X		X		X
FRE55										X	X
FRE56										X	X
FRE57										X	X
FRE58										X	X

4.2.3. CPMS

Acronyms

- **V** View
- **AM** Authentication Middleware
- **LO** Login
- **CS** Charging Station Management
- **EA** eMSP Association
- **EM** eMall API Handler
- **EC** Energy Criteria
- **SO** Set Offer
- **PC** Price Calculation
- **CP** Charging Process
- **BO** Booking
- **PT** Protocol Translator
- **RO** Remote Operation Receiver
- **SU** Station Updates
- **MO** CPMS Model

Table

ID	V	AM	LO	CS	EA	EM	EC	SO	PC	CP	BO	PT	RO	SU	MO
FRE29	X	X	X			X									
FRE30	X														
FRE31	X	X	X	X											X
FRE32	X	X		X								X			X
FRE33	X	X		X								X			
FRE34	X	X		X								X			
FRE35	X	X		X								X			
FRE36	X	X		X								X			
FRE37							X								X
FRE38							X								X
FRE39	X	X					X								X
FRE40	X	X					X								X
FRE41	X	X					X								X

FRE42							X							X
FRE43									X	X				X
FRE44							X							X
FRE45	X	X						X						X
FRE46	X	X			X	X							X	X
FRE47	X	X			X	X							X	X
FRE48											X		X	X
FRE49											X		X	X
FRE50										X			X	X
FRE51										X			X	X
FRE52										X			X	X
FRE53									X	X			X	X
FRE54										X			X	X
FRE55													X	X
FRE56													X	X
FRE57													X	X
FRE58													X	X

5 | IMPLEMENTATION, INTEGRATION AND TEST PLAN

In this section, we specify the order of implementation of each component with a Gantt diagram.

5.1. Implementation and Component Integration

The sequence of component implementation illustrated in the Gantt diagram refers to a relative time and not to an absolute one. All the components can actually be implemented in parallel by using the interface already provided in this document.
In particular, the best idea would be to provide all the methods of the interfaces with a dummy implementation that can be immediately used. After the single component has been implemented it can easily be integrated with the other ones since it already has a defined interface.
If the workforce is not big enough to implement all the components in parallel, the order of implementation should follow the described order of the Gantt diagram, hence, following all the start-to-start cascading relationships.

eMSP Gantt Diagram

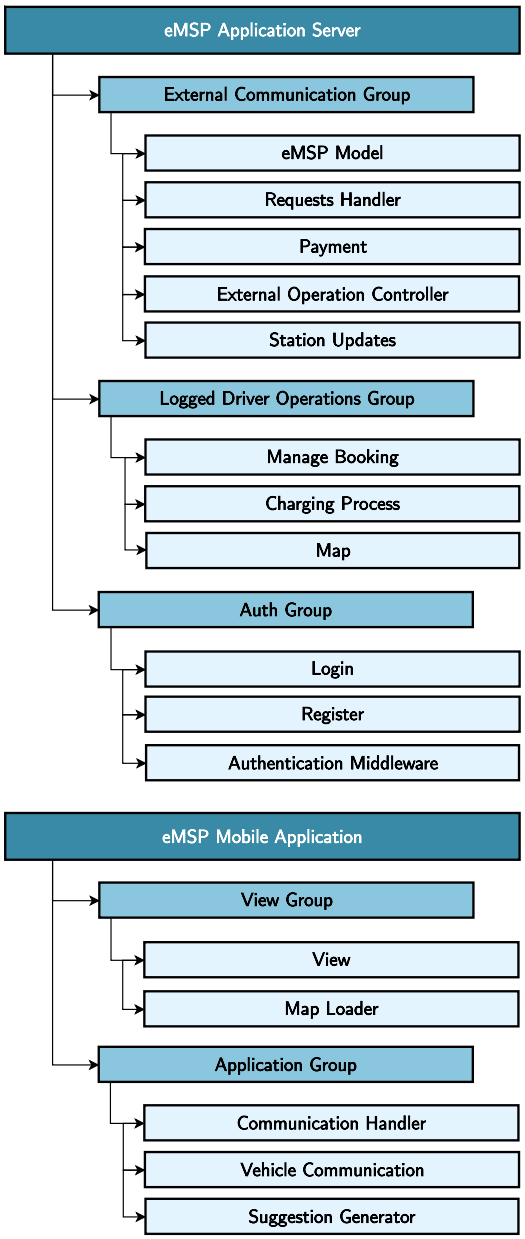

Figure 5.1: eMSP Gantt Diagram

- **External Communication Group:** Components required for communicating with the Driver, APIs and the CPMS.
- **Logged Driver Operations Group:** Components required for processing requests performed by the Driver on the mobile application.
- **Auth Groups**: Components required for the authentication and registration of the Driver.
- **View Group**: Components required for the visualization of the mobile application.
- **Application Group**: Components required for the logic of the user interface.

CPMS Gantt Diagram

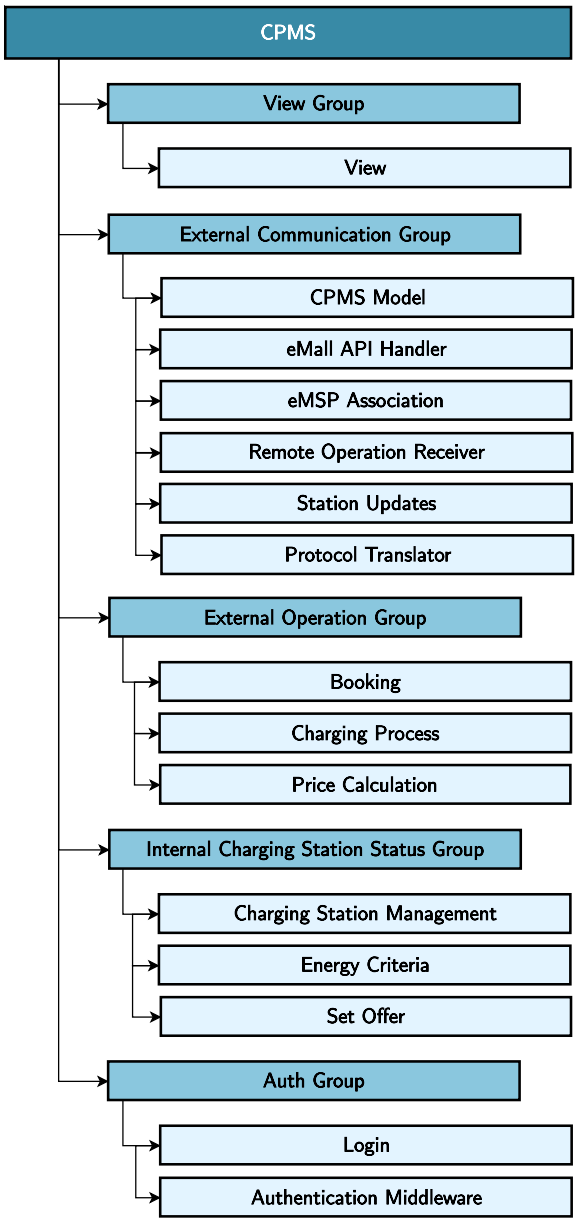

Figure 5.2: CPMS Gantt Diagram

- **View Group:** Components required for the visualization of the user interface.
- **External Communication Group:** Components required for communicating with the eMSP, APIs and stations.
- **External Operation Group**: Components required for processing requests performed by the Drivers through the eMSPs.
- **Internal Charging Station Status Group**: Components required for processing requests performed by the CPO.
- **Auth Groups**: Components required for the authentication and registration of the CPO.

5.2. System Testing

Since all the components could be developed in parallel, a good practice would be to use a TDD approach for the integration testing of the components. This involves writing a test for all interface methods of each component as the interface is being developed. These tests will initially fail and only pass once the component has been implemented. In addition, unit testing should be performed for each component after implementation. This means that the testing phase will be integrated throughout the entire implementation process, rather than just at the end. Overall we can split the testing phase into different parts:

1. **Interfaces test writing:** Write tests for each interface method. All of them should fail.
2. **Integration testing:** After a single component is developed and has replaced its stub, it should pass the test for its external interface.
3. **Unit testing:** After a single component is developed, the developer should write tests in order to assure that every function inside that component is working in the proper way.
4. **Performance testing:** After all the components have been developed and integrated, then there should be performance testing to make sure that there aren't any bottlenecks.
5. **Load testing:** Load testing should be done in order to avoid bugs such as memory leaks or buffer overflows. There is also the possibility to test the system in order to understand how long it can withstand the maximum load.
6. **Stress testing:** Stress testing is used to make sure that the system recovers carefully from a failure.
7. **Networking testing:** Networking testing is used to make sure that the system can maintain its behaviour in case of a down node.
8. **Acceptance testing:** This part should be the last part of the project, done with the stakeholders. It should check if the project respects the requirements and if the system satisfies the assignment.

6 | EFFORT SPENT

Here is reported the amount of time spent writing this document.

Andrea Piras	Hours
Introduction	2
Architectural Design	36
User Interface Design	10
Requirements Traceability	12
Implementation, Integration and Test Plan	9
Total	69

Emanuele Santoro	Hours
Introduction	1
Architectural Design	35
User Interface Design	11
Requirements Traceability	16
Implementation, Integration and Test Plan	6
Total	69

Andrea Sanguineti	Hours
Introduction	1
Architectural Design	49
User Interface Design	13
Requirements Traceability	1
Implementation, Integration and Test Plan	5
Total	69

7 | REFERENCES

7.1. Reference Documents

- Specification document: Assignment RDD A.Y. 2022-2023.
- Requirement Analysis and Specification Document (RASD).
- Course slides

7.2. Reference Sites

- `Diagrams WebSite`
- `Sequence Diagrams WebSite`
- `Overleaf WebSite`
- `Online Visual Paradigm WebSite`
- `Lucid App WebSite`
- `WebHook WebSite`
- `MVVM pattern`
- `JWT`
- `JPA`