DD

Software Engineering Project
A.Y. 2022-2023

Marco Ronzani, Alessandro Sassi

December 2022

Contents

1 Introduction

1.1 Purpose

Electric vehicles are starting to grow in number, and their takeover of combustion engines is bound to happen, consequently to support such a thriving trend adequate easy access to charging stations is of utmost importance. In this landscape the goal of eMall is to allow owners of electric vehicles to easily know where charging stations are and carefully plan their charging process according to their schedules at any such station.

This document will follow up on the RASD document with a discussion of the architectural design of the system, its main architectural components and their interfaces. It will also cover the plans for implementation and integration, as well as those for testing, with the goal of guiding the development process.

1.2 Scope

This DD document takes into consideration the requirements and specifications of the eMSP platform "eMall", together with its interaction with one or more CPMSs. The stakeholders considered are the End Users who interact with the "eMall" platform, CPOs owning the respective CSs and CPMSs, and DSOs offering their services to the aforementioned parties.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

Term	Definition
Charging Station	Device with a connection to the electric grid which brakes out power to one or more socket(s) for vehicles. Monitoring of each socket's status.
Socket	One of the charging outlets available at a CS where a vehicle connects, its type determines the vehicles that can connect.
Charging Session	The process in which a User performs a recharge of their vehicle at a specific socket.
Energy source of a Charging Station	Batteries or the DSO currently assigned to the CS, whichever the CS is currently drawing its power from.
Charging Station External Status	Number of charging sockets available, their type such as slow/fast/rapid, their cost, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed.
Charging Station Internal Status	Amount of energy available in the batteries, if any, number of vehicles being charged and, for each charging vehicle, amount of power absorbed and time left to the end of the recharge.
Energy Source	Method of energy production that results in a known fraction of the energy supplied to an endpoint in the electric grip.
User-price	Cost of a recharge that is shown to the User when they inspect a CS and is what they are charged for. It is set by the CPO/CPMS on a per-CS basis.
Nominal-price	Cost of a recharge at a CS without any discount or offer applied, it is set by the CPO/CPMS on a per-CS basis. A user-price $<$ nominal-price implies an ongoing special offer.
Energy source management policy OR Battery usage policy	A per-CS policy given them by the CPMS to allow them to dynamically decide whether to acquire energy from their assigned DSO or from their batteries and when to charge their batteries with energy from their DSO.
Charge Point Management System's policy for "Automatic Mode"	The global policy used by the CPMS to operate autonomously, its mainly built around thresholds and weights to allow the CPMS to decide prices and battery usage policies for its CSs.
Offer reset date	The date, set on a per-CS basis, at which a CS will reset its user-price to the value of its nominal-price, removing any present offer.

1.3.2 Acronyms

Acronym	Full Name
eMall	Electric Mobility for All
eMSP	Electric Mobility Service Provider
CPO	Charging Point Operator
CPMS	Charge Point Management System
CS	Charging Station
DSO	Distribution System Operator
STB	System-To-Be
OS	Operating system
DB	Database

1.4 Revision History

v0.1 First draft of the document.

1.5 Reference Documents

1. The provided document describing the project: *Assignment RDD AY 2022-2023-v3.*
2. The Software Engineering 2 course held by Prof.s Camilli Matteo, Di Nitto Elisabetta and Rossi Matteo Giovanni.
3. *ISO/IEC/IEEE 29148:2011(E)* standard for Requirement Engineering.
4. Project of last year provided as an assignment.

1.6 Document Structure

The **architecture design** introduces the structure of the STB and its main components in regard to both the eMSP and the CPMS. Every component is thoughtfully described in its purpose and position in the components hierarchy. Following the components description in a view of how the STB will be deployed and its runtime interactions between the various components of across both the eMSP and CPMS. In the subsection the main design decision behind the system architecture are elicitated and discussed in futher details.

The **user interface design** section resumes what was already present in the RASD document and futher expands on it.

The **implementation, integration and test plan** section then proceeds to discuss how the architecture previously described will be realized, with its modules integrated and tested in order to guarantee the correctness of the final product, w.r.t. the requirements and directions expressed both in this document and the RASD, during each step of development.

The **effort spent** section presents data regarding the amount of time each team member invested in the creation of this document.

Finally **references** are reported.

2 Architectural Design

2.1 Overview

The system architecture can be split into 2 main parts: one for the eMSP, the other one for the CPMS. The eMSP adopts an architecture organized into **four layers** distributed over **four tiers**, resulting in a monolithic structure with benefits for security and maintenance, given that there are less entry points for possible attack vectors. This monolityic architecture is also quite appropriate for the system-to-be, since the services it needs to provide are all closely related. However, it should be noted that its modular approach would allow for an easy transition to microservices when, if ever, the load imposed on the system will require the distribution of its functionalities over multiple machines. Therefore, the eMSP system is composed of a **presentation layer**, running the client-side application, a **Web Server** and **Application Server** to handle API routing and web app page serving, and finally a **dedicated DB Server**.
The CPMS adopts an analogous architectural pattern, with the same **four layers** and **four tiers**. However, in order to communicate with the CSs it owns, they are modeled as clients, which run a custom application and communicate with the CPMS platform through the Web Server and its exposed APIs.

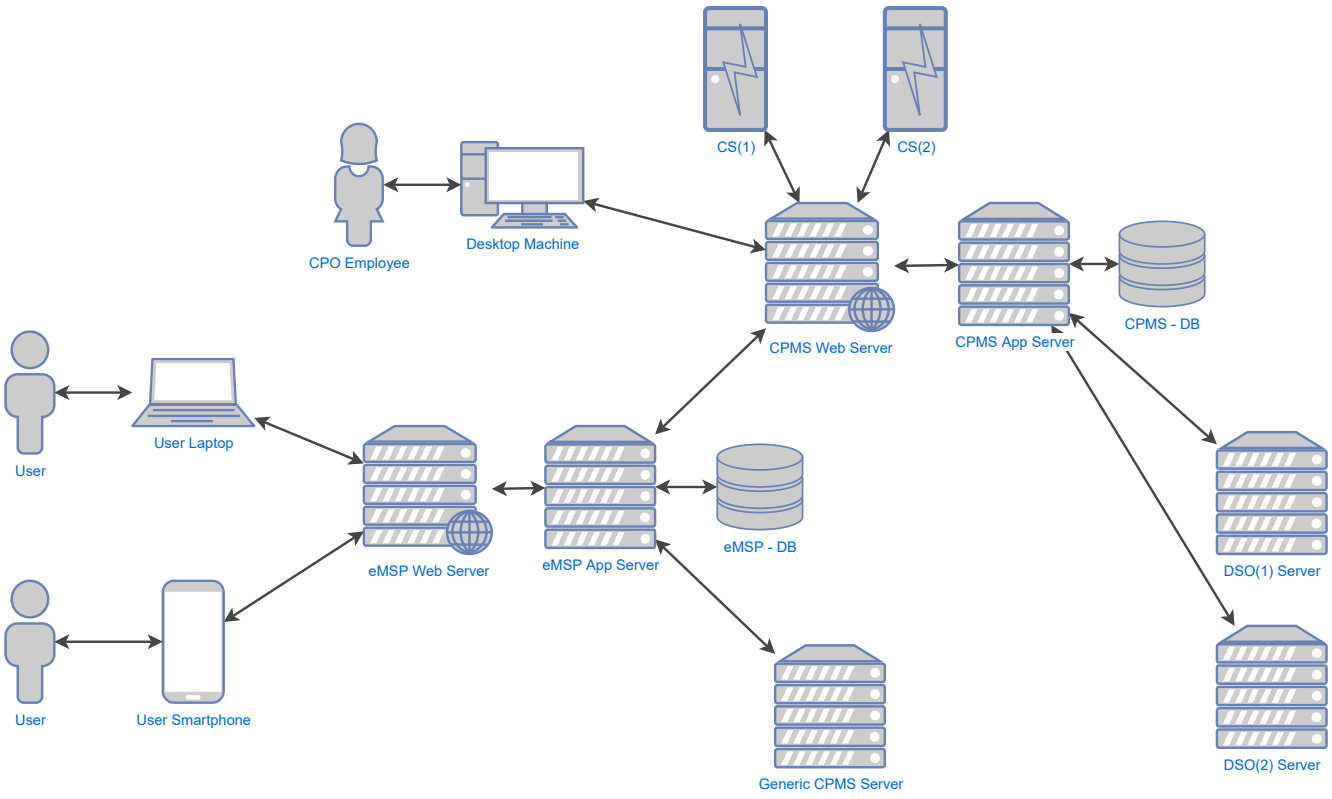

Figure 1: Overview Diagram

- **eMSP Web Application**
  A Web Application that runs within the User's web browser and that gives them access to the eMall services. All the resources it needs get downloaded to the User's browser from the eMSP Web Server as the the device reaches it through HTTP requests.

- **eMSP Web Server**
  Back-end component that interfaces the User's web browser with the other application services in the eMSP's back-end. It handles HTTP requests arriving from user-devices by routing them to the corresponding components that can respond to them.

- **eMSP App Server**
  The principal back-end component for the eMSP, it contains all the modules that together offer all functionalities available via the eMSP. Requests are routed from the Web Server to the module capable handling them, the response is than sent from the App Server through the Web Server to the User's device. The App Server is also capable of accessing the eMSP DB and the external services needed for its functions.

- **eMSP DB Server**
  Component dedicated to data storage for the eMSP, it stores for instance the Users' credentials and bookings, to ensure its safety it can only be accessed by the eMSP's App Server.

- **CPMS Web Application**
  A Web Application meant for the desktop devices available to CPO Employees, it runs withing the browser and allows its users to access the management functions of the CPMS. It dialogues with the CPMS Web Server through the HTTP protocol.

- **CPMS Web Server**
  Back-end component that interfaces the CPO Employees' web browser and the back-ends of the eMSP with the application services in the CPMS's back-end. It handles HTTP requests arriving from the outside by routing them to the corresponding components that can respond to them and allows tunneling of WebSocket connections from CS to the back-end.

- **CPMS App Server**
  The principal back-end component for the CPMS, it contains all the modules that together offer all functionalities available to the CPO and eMSPs, as well as being the endpoint where CS connect to in order to be managed. Requests and connections are routed from the Web Server to the module capable handling them, in the fist case the response is than sent from the App Server through the Web Server to the requesting device, in the second case the connection is handled and kept alive by its target module. The App Server is also capable of accessing the CPMS DB and the external services needed for its functions.

- **CPMS DB Server**
  Component dedicated to data storage for the CPMS, it stores for instance the credentials for CPO Employees as well as a local copy of the CSs' configurations, to ensure its safety it can only be accessed by the CPMS's App Server.

- **CS**
  External entity w.r.t. the STB, it is configured to reach out to the CPMS App Server via a WebSocket connection that allows the CPMS to manage and monitor it.

- **DSO**
  External entity w.r.t. the STB, it exposes a web API that allows CPMSs to acquire information regarding its price and energy mix.

- **Maps Provider**
  External entity w.r.t. the STB, offers to clients up to date maps of any requested area, as well as related information such as satellite images.

2.2 Component View

In this section, every major component is analysed in terms of its sub-components. Everything that is outside of the system is considered as a black box.

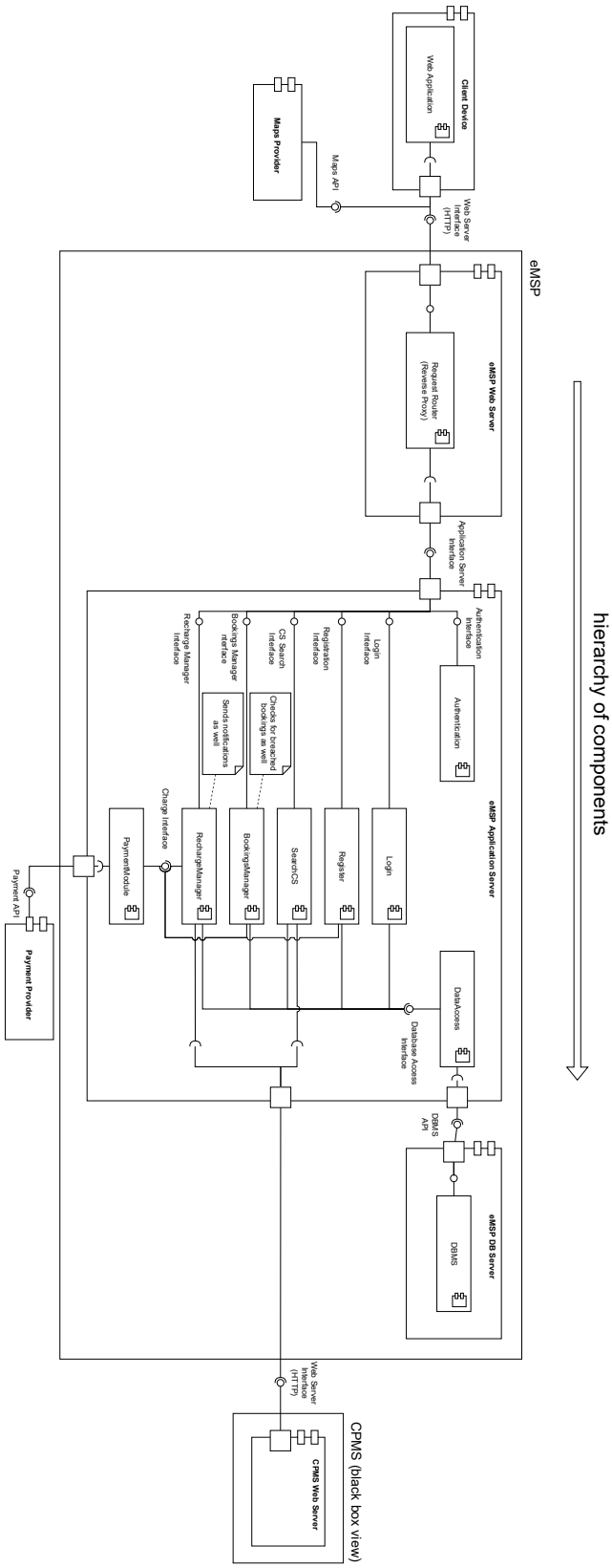

Figure 2: Component diagram

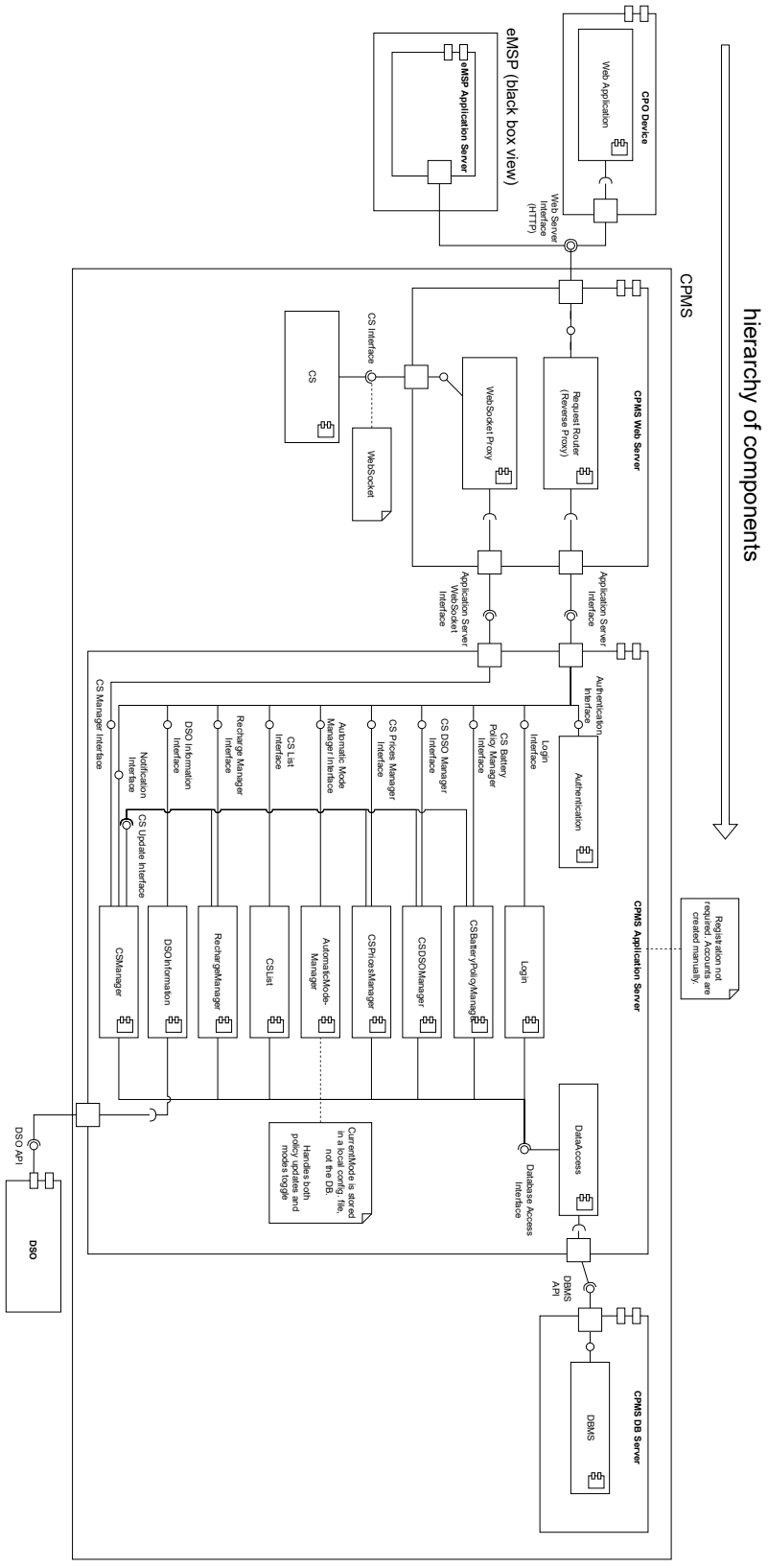

Figure 3: Component diagram

2.2.1 Client Device

Client devices represent all the devices connected to the internet and used to reach the eMall website. By reaching eMall the **eMSP Web Application Component** in the form of an interactive web application is loaded by the device's web browser and eMall's UI is shown to the User.

The **eMSP Web Application Component** is the front-end that it offers to its Users, it consists of a website offering an interactive web application with a UI that allows Users to access every function offered by eMall. This component does not integrate computational intensive tasks and in itself does not have any data that can instead be provided by eMall's back-end. Its functionalities are limited to error checking, UI rendering and contacting the eMSP's back-end to recover any data needed to offer the User the intended functionalities.

2.2.2 eMSP Web Server Component

The eMSP Web Server is required in order to serve the eMSP Web Application to User, and to allow such Web App to interact with the eMSP's back-end when needed.

To this end its main sub-component is a **Request Router**, a Reverse Proxy component that sits in front of the back-end and routes requests from client devices to the back-end services capable of satisfying them, this way the resources returned to clients appear as if they originated from the Web Server itself. The role of the Reverse Proxy is also to help increase scalability, performance, resilience and security by both acting as a content cache and as a middleware before the back-end.

2.2.3 eMSP Application Server Component

The back-end for the eMSP, as per the hierarchy of components, every request reaching it is first checked for an active session, and only those with one which has a performed valid login are allowed access to components other than **Register**.
Every component present performs all the required checks on the received data to correctly perform any of its functions.

- **Authentication**
  Sub-component that handles the authentication by checking the credentials for every request reaching the back-end.
- **Login**
  Sub-component which checks the login credentials inserted by Users through the Web App and grants them an authenticated session if those are valid.
- **Register**
  Sub-component that allows Unregistered Users to create a new account with eMall.
- **SearchCS**
  Sub-component whose purpose is to allow authenticated Users to query eMall for the CSs it has available with the use of filters. It returns the list of matching CSs.
- **BookingsManager**
  Sub-component offering the functionalities to both create a new booking and receive the current list of bookings for the authenticated User, with the possibility of deleting any booking.
- **RechargeManager**
  Sub-component which allows for control over a booking which is currently active, after a User has

connected their vehicle to a booked CS, via this component they can start/stop the charging process as well as get notified of the process terminating.

- **PaymentModule**
  Sub-component that interfaces with an external payment provider and allows the eMSP to charge Users for any obtained service or fee if they were not to show up for a booking.
- **DataAccess**
  Sub-component that provides access to the DB for every other sub-component.

2.2.4 CPO Device

CPO Devices represent all the machines available to CPO Employees to operate with the company's CPMS.

The **CPMS Web Application Component** is meant to be ran inside a web browser on the CPO Devices and consequently be available to CPO Employees, it is the front-end offered by the CPMS and its UI is meant to offer to its users all the management functionalities of the CPMS. This component does not integrate computational intensive tasks and does not store locally any data that can instead be provided by the CPMS's back-end. Its functionalities are limited to error checking, UI rendering and contacting the CPMS's back-end to recover any data required for its functionalities.

2.2.5 CPMS Web Server Component

The CPMS Web Server is required in order to serve the CPMS Web Application to CPO Employees, to allow such Web App to interact with the CPMS's back-end and to allow CS to reach the CPMS with a WebSocket connection, allowing the CPMS to actively manage them.

To this end its main sub-components are:

- **Request Router**
  A reverse proxy that sits between a client and the CPMS back-end, its role is to route requests from clients to the back-end services capable of answering them, this way the resources resources returned to clients appear as if they originated from the Web Server itself. The role of the Reverse Proxy is also to help increase scalability, performance, resilience and security by both acting as a content cache and as a middleware before the back-end.
- **WebSocket Proxy**
  Its role is to tunnel WebSocket connections between the CPMS back-end and the CSs, after those are originated from upgrade requests over HTTP that went through the Request Router.

2.2.6 CPMS Application Server Component

The back-end for the CPMS, every request reaching it is let through to the CPO-only components only if it has an active authenticated session with CPO credentials. Access to components that are meant for eMSPs is granted to all eMSPs that are known by the system as long as their request are correctly performed with an authenticated session.
Every component present performs all the required checks on the received data to correctly perform any of its functions.

- **Authentication**
  Sub-component that handles the authentication by checking the credentials for every request reaching the back-end.

- **Login**
  Sub-component which checks the login credentials inserted by CPO Employees through the Web App and grants them an authenticated session if those are valid.
- **CSBatteryPolicyManager**
  Sub-component that allows CPOs to change the policy currently in use by a CS to manage its batteries, it both saves to the CPMS's DB updated CS policies as well as requests the CSManager to communicate the updated policies to the actual CS.
- **CSDSOManager**
  Sub-component that allows CPOs to change the DSO currently supplying a CS, it both saves to the CPMS's DB updated CS's DSO as well as requests the CSManager to communicate the change to the actual CS.
- **CSPricesManager**
  Sub-component that allows CPOs to change the user-price, the nominal-price and the offer reset date of a CS, it both saves to the CPMS's DB updated prices and date as well as requests the CSManager to communicate the updated values to the actual CS.
- **AutomaticModeManager**
  Sub-component which allows CPO Employees to toggle the operating mode of the CPMS from "Automatic Mode" to "Manual Mode" or vice-versa, as well as allowing the CPO to update the CPMS's policy for "Automatic Mode".
- **CSList**
  Sub-component providing the CPO and external eMSPs with the list of CS currently under the control of the CPMS. It also accepts filters to reduce the list of provided CSs. If they are requested, details regarding a specific CS can also be provided by this component.
- **RechargeManager**
  Sub-component providing external eMSPs with the possibility of managing, on behalf of their Users, the ongoing charging process to a certain socket of a CS. This module also provides the eMSP feedback on the current status of the charging process. This component dialogues with the CSManager to gather the CS status and forward it commands to manage the charging process.
- **DSOInformation**
  Sub-component providing the CPO with information regarding the DSOs known by the CPMS. This module also interfaces directly with the DSOs external API to gather their prices and energy mixes.
- **CSManager**
  Sub-component that allows the other back-end components to dialogue with CSs, know their real-time status, coordinate their charging processes and update their configuration. CS signal this module when one of their recharges is complete, and this module forwards such notification to the eMSP. This module also accepts WebSocket connections from CS to allow their management.
- **DataAccess**
  Sub-component that provides access to the DB for every other sub-component.

2.2.7 eMSP DB Components

SQL DB for the eMSP, it contains the accounts and bookings of all users.

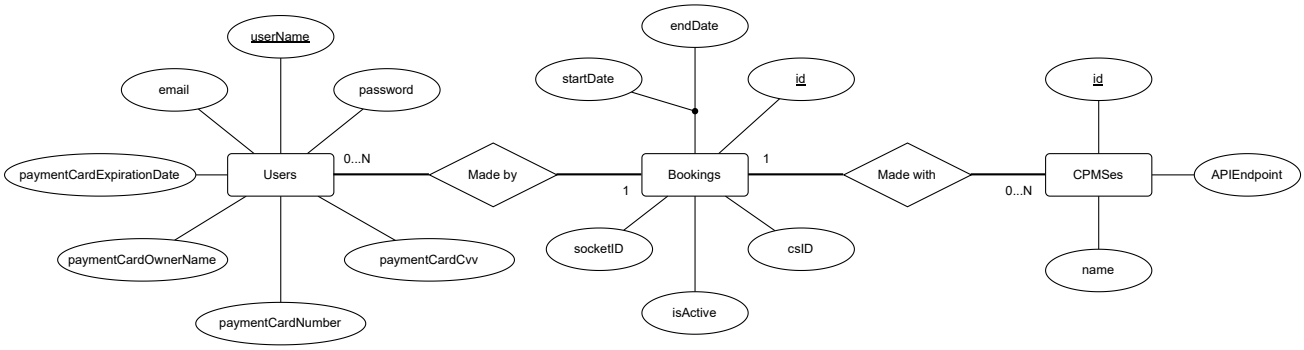

Figure 4: eMSP ER Diagram

2.2.8 CPMS DB Components

SQL DB for the CPMS, it contains information on all known CSs and their configurations.

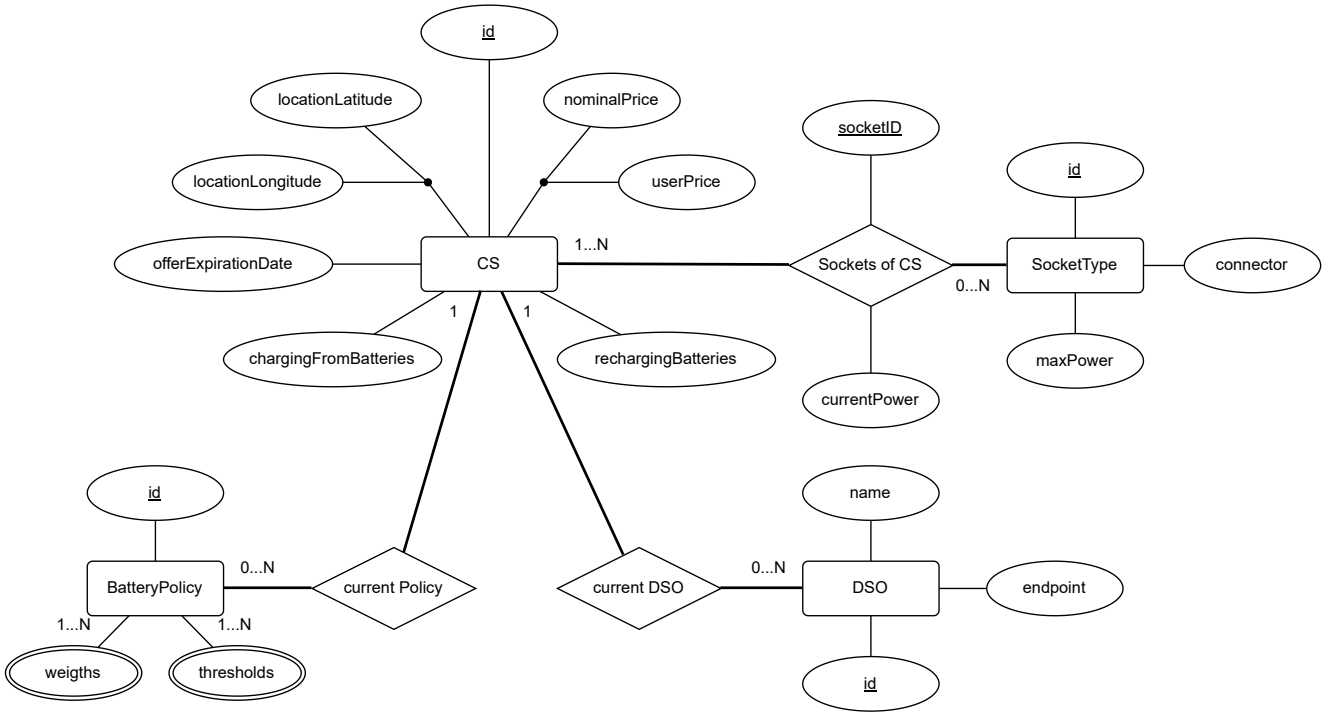

Figure 5: CPMS ER Diagram

2.2.9 External Systems

Systems outside the scope of this document, whose existence is acknowledge in order to explain the behaviour of certain components of the STB.

- **DSO**
  An endpoint with an exposed API that allows for retrieval of information regarding the DSO's energy mix and prices.
- **Payment Provider**
  An endpoint that allows, given adequate payment details, to charge someone for a service and move the sum to a certified bank account.
- **Map Provider**
  A service providing up to date maps for the entire world. It is reached by the eMSP Web Application to improve the UI with an interactive map.

2.3 Deployment View

Here we describe the arrangement of physical nodes and the components deployed on them. **Load balancers** and **Firewalls** are also included in the diagram:

- **Firewall**
  Is directly connected to the Internet and filters incoming packets, only letting pass those identified as trustworthy.
- **Load balancer**
  A device that forwards requests to the back-end's services in accordance to their current load, aiming to distribute work evenly. Its presence improves the maximum load capacity the system can handle and its reliability.

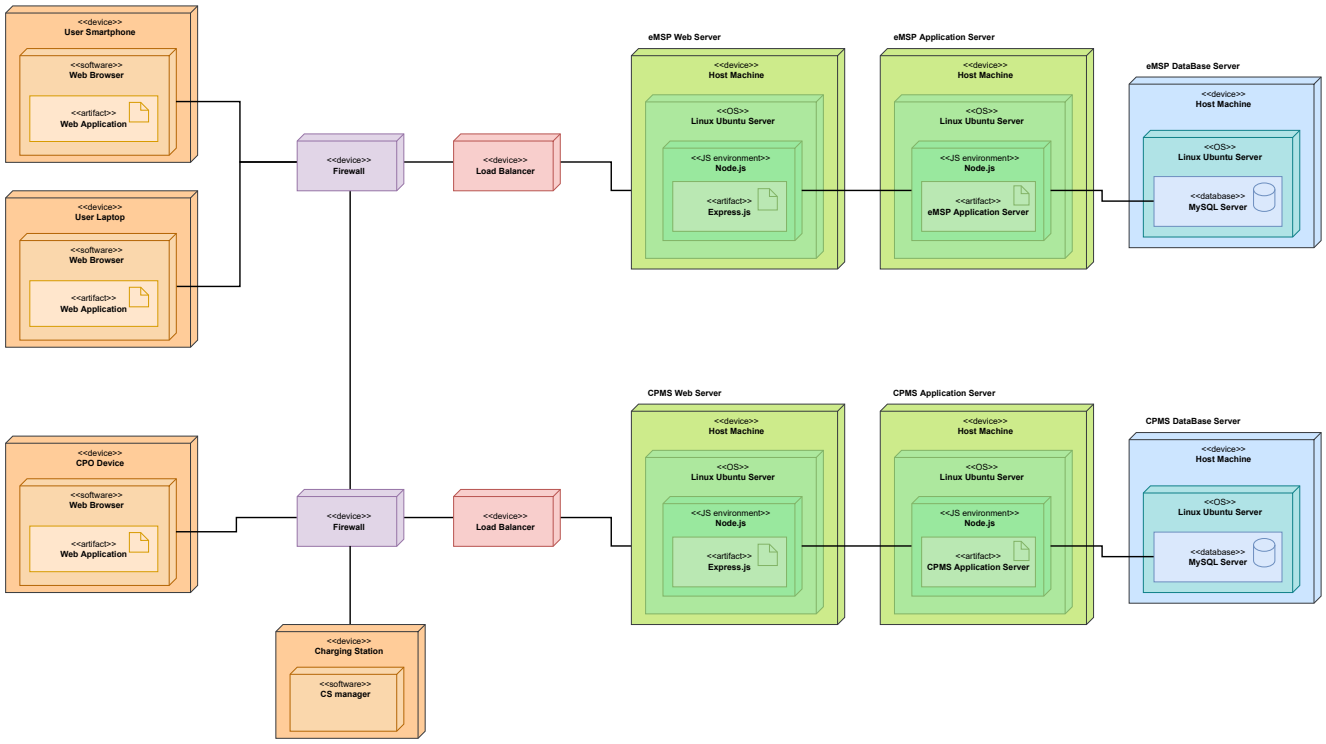

Figure 6: Deployment View

Client devices will only need to reach the Web Server of the eMSP, which will then forward their requests to the eMSP's back-end if needed.

2.4 Runtime View

In this section the dynamic behaviour of the system is presented via sequence diagrams that summarize all the possible interactions that can be had with both the eMSP or the CPMS. Each diagram further expands on those discussed in the RASD document in section 3.2.3 .

1. New user registration

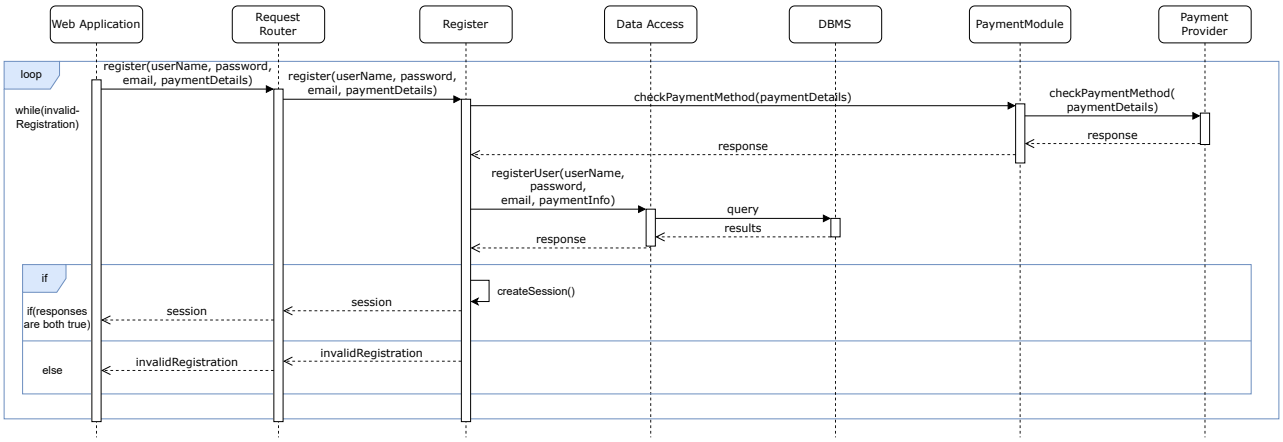

Figure 7: New user registration

2. User login

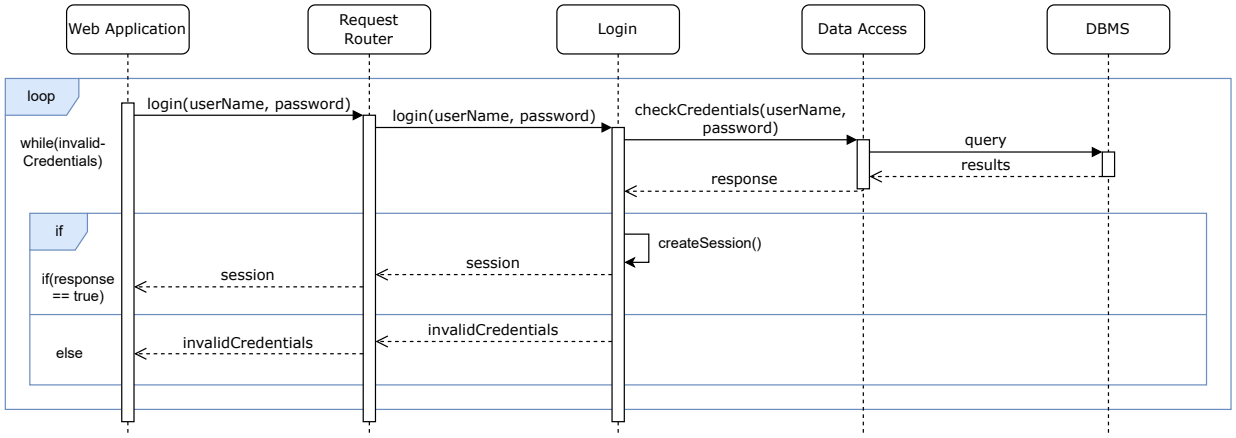

Figure 8: User login

3. User searching and booking a CS

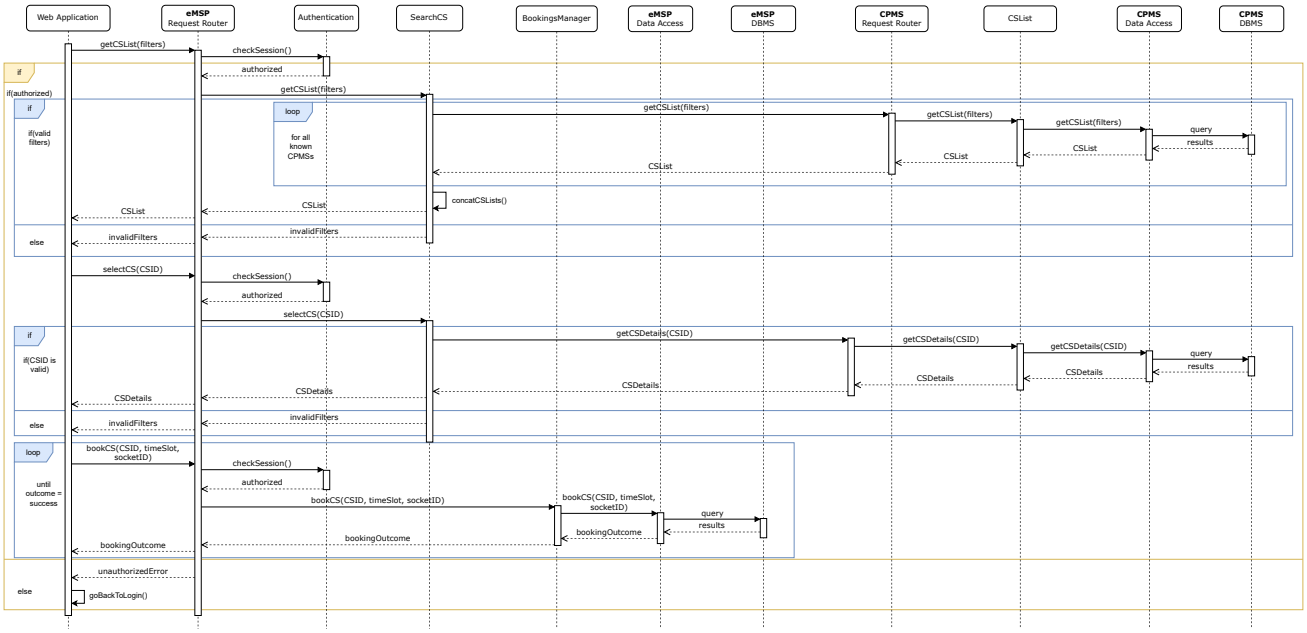

Figure 9: User login

4. User deleting one of his bookings

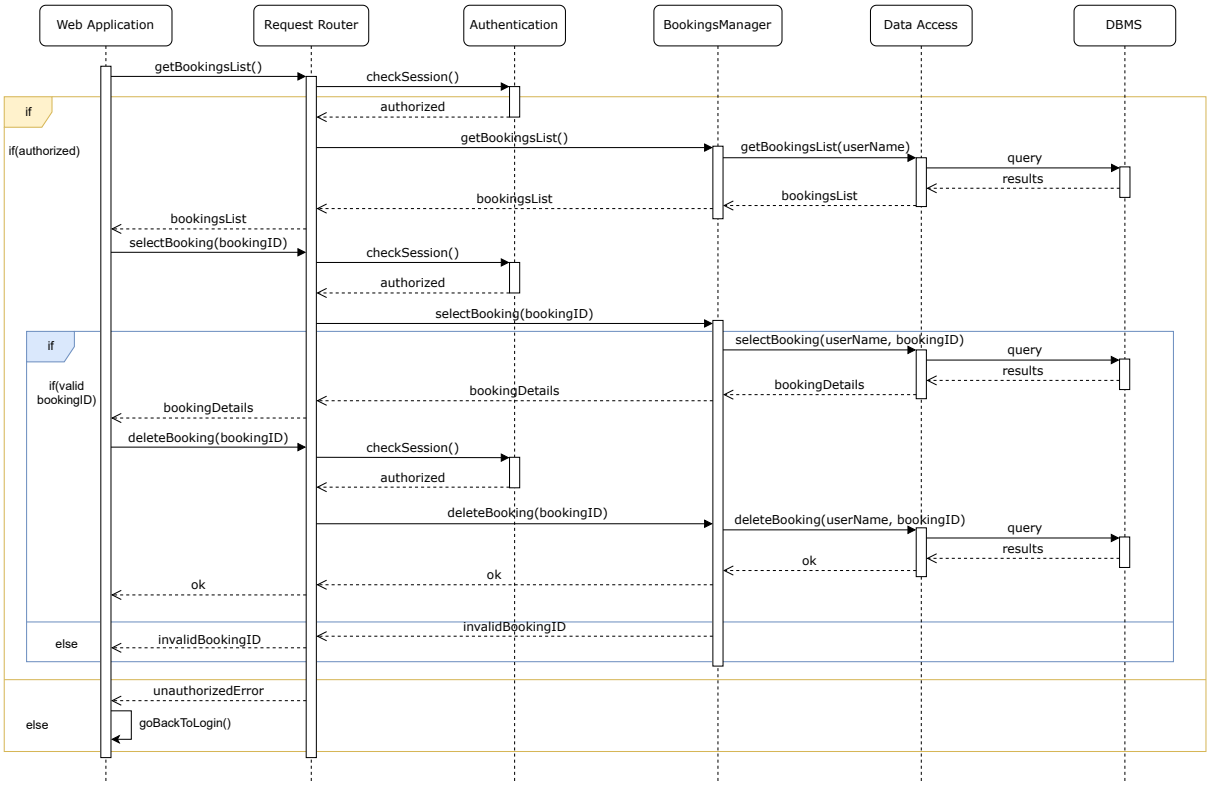

Figure 10: User deleting one of his bookings

5. User starting a charging process

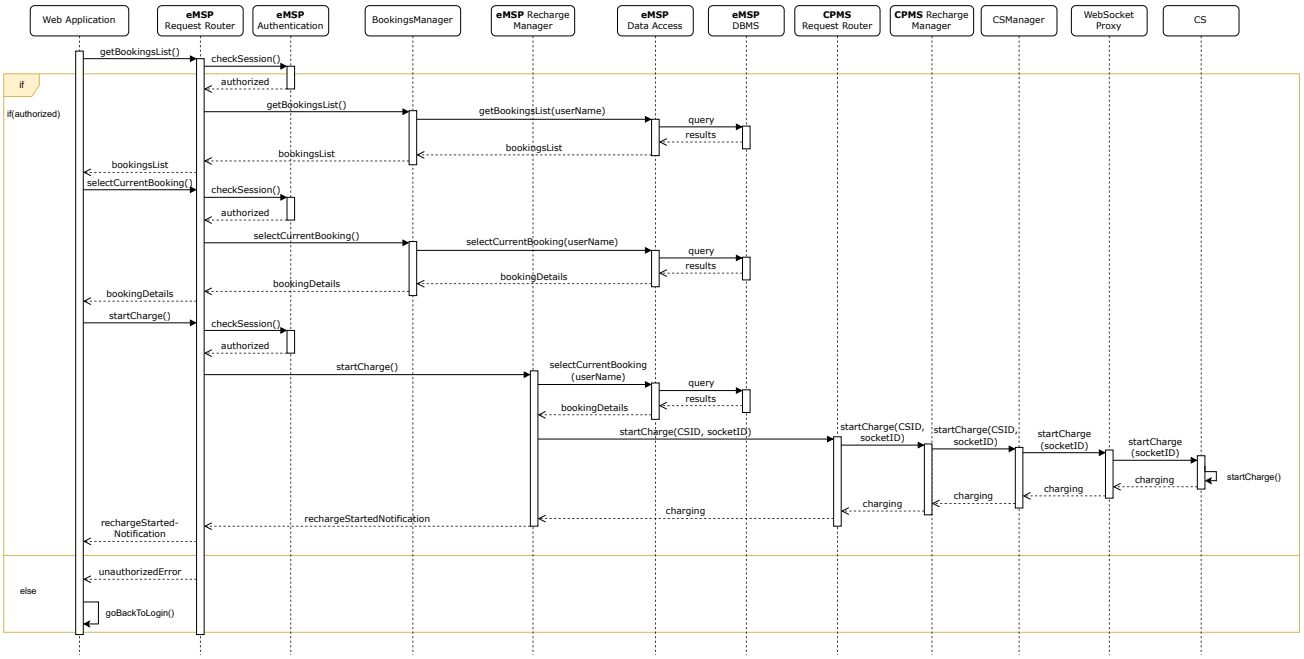

Figure 11: User starting a charging process

6. User interrupting a charging process

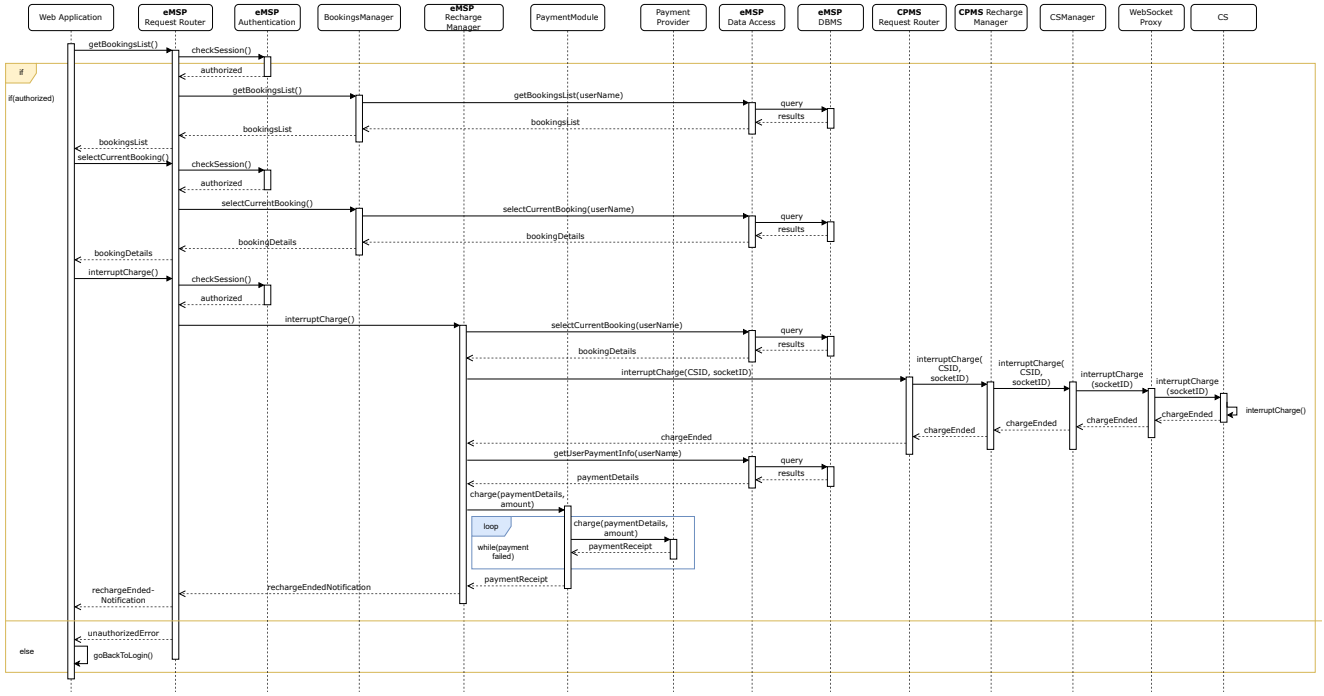

Figure 12: User interrupting a charging process

7. Charging procedure self-terminates and User receives a notification

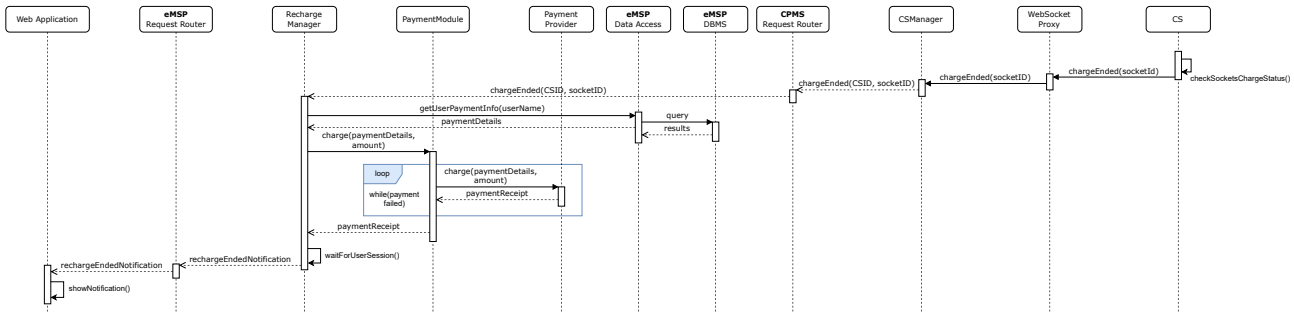

Figure 13: Charging procedure self-terminates and User receives a notification

8. User does not show up for a booked recharge

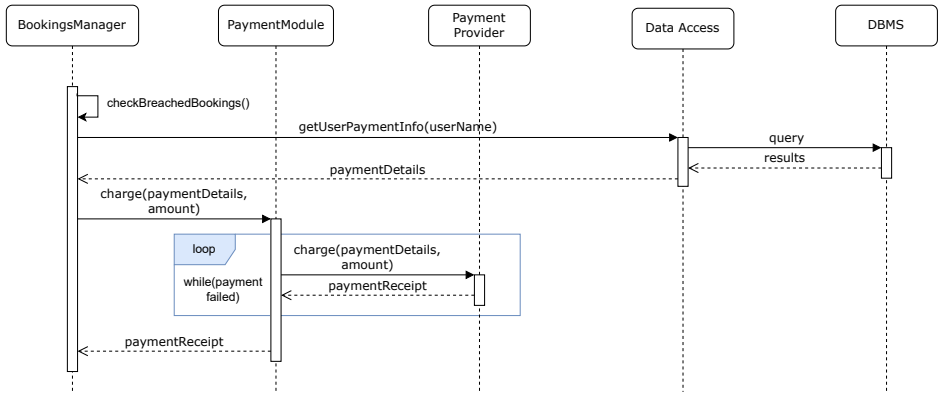

Figure 14: User does not show up for a booked recharge

9. CPO logs in into the CPMS

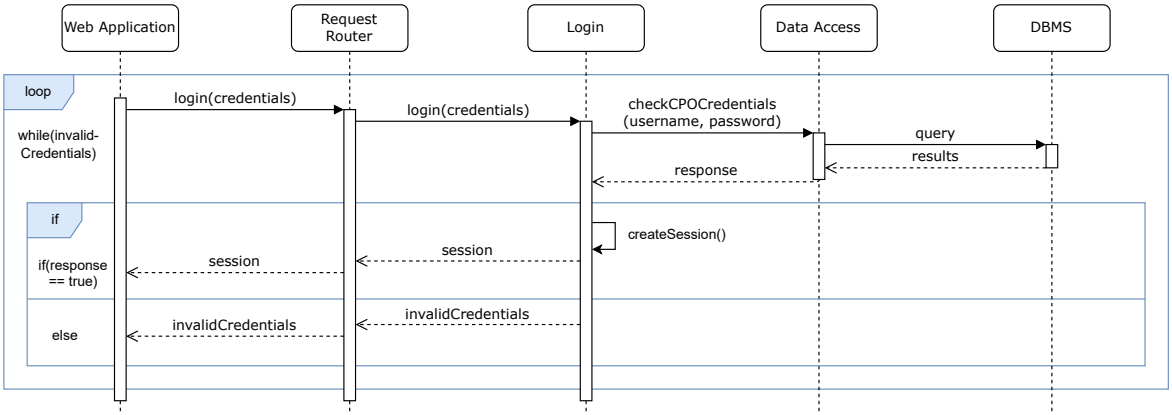

Figure 15: CPO logs in into the CPMS

10. CPO retrieves energy prices and mixes from DSOs

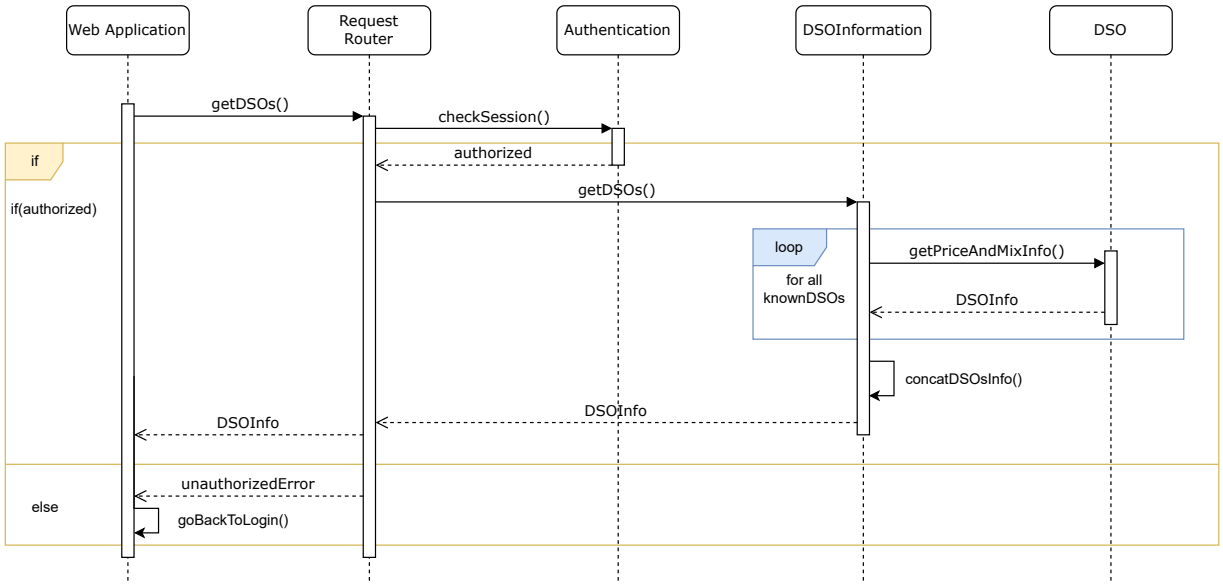

Figure 16: CPO retrieves energy prices and mixes from DSOs

11. **CPO assigns nominal-price, user-price, energy sources and battery usage policies for a CS**

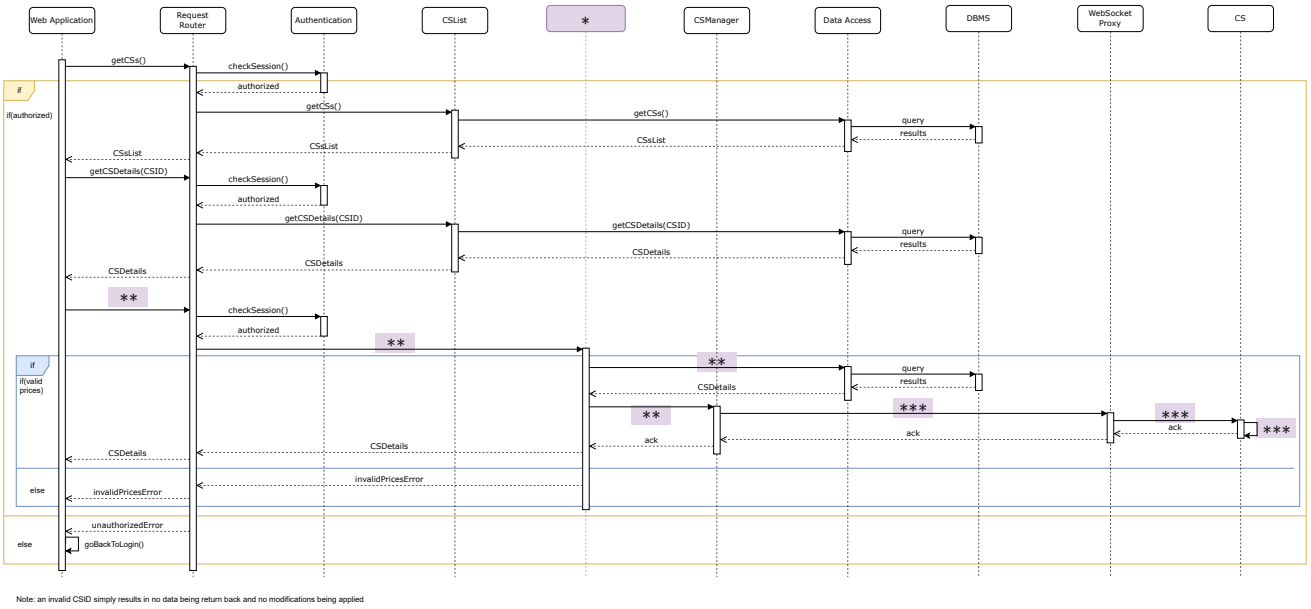

Figure 17: CPO assigns nominal-price, user-price, energy sources and battery usage policies for a CS

This diagram summarized 3 views in one, by simply substituting the following components and interactions to the $*s$ one can obtain the different versions:

1. **CPO assigns nominal-price, user-price**

   $*$ $\rightarrow$ CSPricesManager

   $**$ $\rightarrow$ updateCSPrices(CSID, userprice, nominalprice)

   $***$ $\rightarrow$ updateCSPrices(userprice, nominalprice)

2. **CPO assigns energy sources**

   $*$ $\rightarrow$ CSDSOManager

   $**$ $\rightarrow$ updateCSDSO(CSID, DSOId)

   $***$ $\rightarrow$ updateCSDSO(DSOId)

3. **CPO assigns energy battery usage policies**

   $*$ $\rightarrow$ CSBatteryPolicyManager

   $**$ $\rightarrow$ updateCSBatteryPolicy(CSID, thresholds, weights)

   $***$ $\rightarrow$ updateCSBatteryPolicy(thresholds, weights)

12. CPO toggles the CPMS operating mode

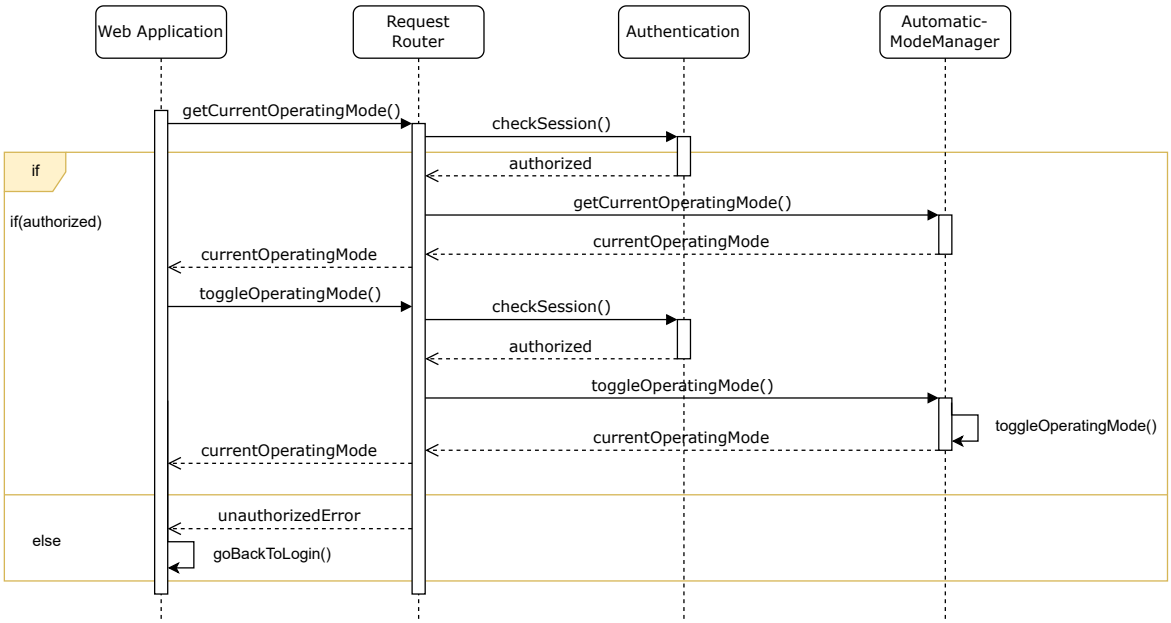

Figure 18: CPO toggles the CPMS operating mode

13. CPO changes the CPMS's "Automatic Mode" policy

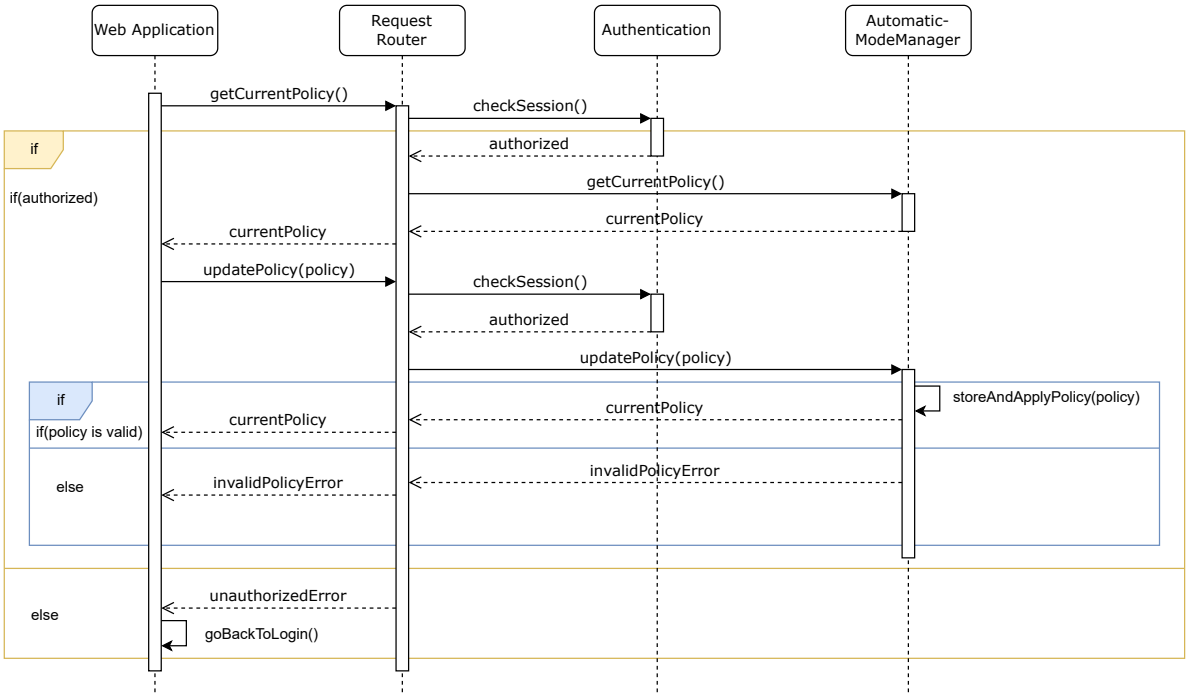

Figure 19: CPO changes the CPMS's "Automatic Mode" policy

2.5 Component Interfaces

This section provides a summary of the interfaces required and offered by the various components of both the eMSP and CPMS in the form of the following diagram.

Figure 20: Interfaces diagram

2.6 Selected Architectural Styles and Patterns

Regarding architectural styles, both systems will use a **Client-Server** architecture, where clients can be both consumer-accessible devices (e.g. laptops, desktop PCs) or, as is the case for the CPMS, internal CS software. For communication, **REST APIs** are extensively used to expose a standard interface to all clients involved in the process, while adopting a de-facto industry standard for client-server communication.
To support the aforementioned client-server communication, the **HTTP** protocol is used, except for messages exchanged between the CSs and CPMS, which adopt the **WebSocket** protocol.
A **Model-View-Controller** approach is adopted for both the eMSP and CPMS infrastructures. The three components can be mapped as follows:

- **View**: The Client-side HTML/CSS code, as well as JavaScript views used to present data to the user;
- **Controller**: The Client-side modules used to asynchronously request and send data to the back-end, but also the Server-side components handling all API routes;
- **Model**: Server-side representations of the data stored in the DB;

Therefore, the client can be classified as a **Fat Client**, since it incorporates part of the controller logic to handle asynchronous client-to-server communication and view updates.
This architectural style also maps easily to the **4-tier** architecture used by both systems, where the **Presentation Layer** includes both the **View** and part of the **Controller** (due to the fat client approach), the **Web Server** and **Application Server** run all the server-side **Controller** modules, as well as abstract **Model** representations, and the **DB Server** manages all the **Model** data.
Finally, the common **Observer** pattern is adopted for asynchronous events that require an action or a response by either a client or server. This pattern is employed mostly to handle CS event notifications, as well as for managing events such as a user skipping one of their bookings.

2.7 Other Design Decision

For security reasons, all passwords are never stored as clear text, but rather encoded using standard hashing algorithms (Ex: Bcrypt).

3 User Interface Design

The interface eMall will use is a web app, through which its Users will have available all product functions. Such web app will need to be usable both on desktop and mobile devices, since a mobile device is what users will most likely have available to them while at the CS.
In order to allow the User to select an area of interest while searching for CSs, the user interface will make use of a map.

What follows are some mockups for the eMSP's user interface design.

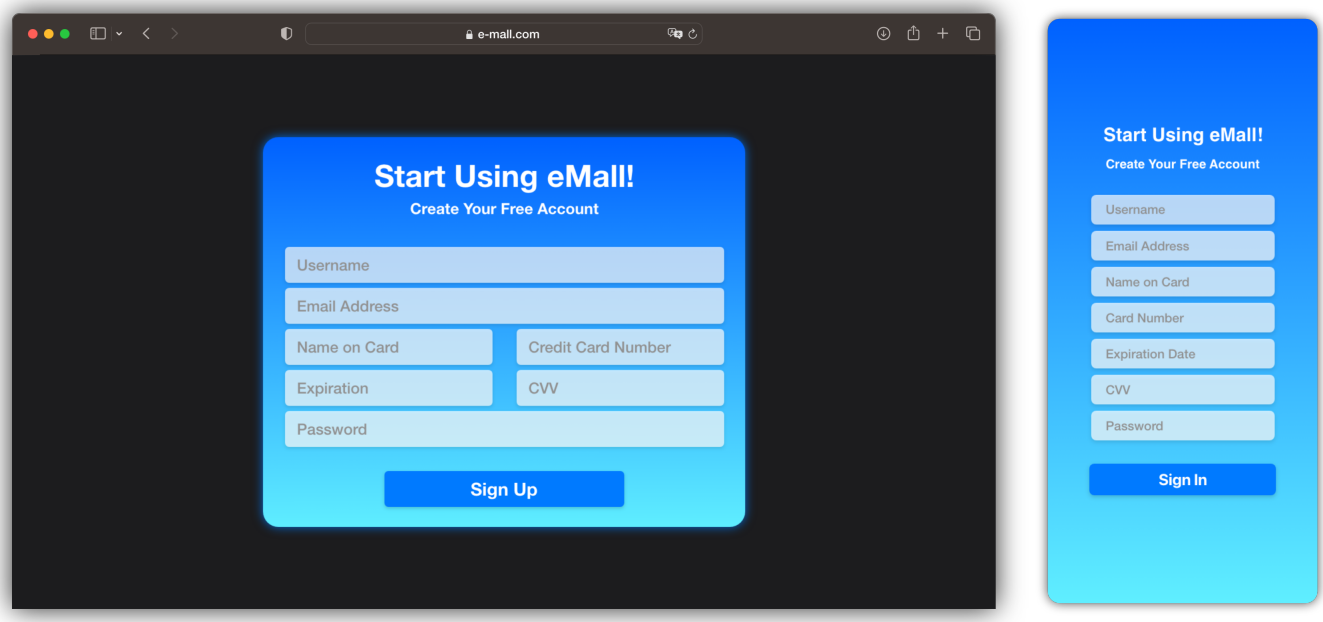

Figure 21: Signup page mockup
Users will be able to register into eMall by compiling the form with their data

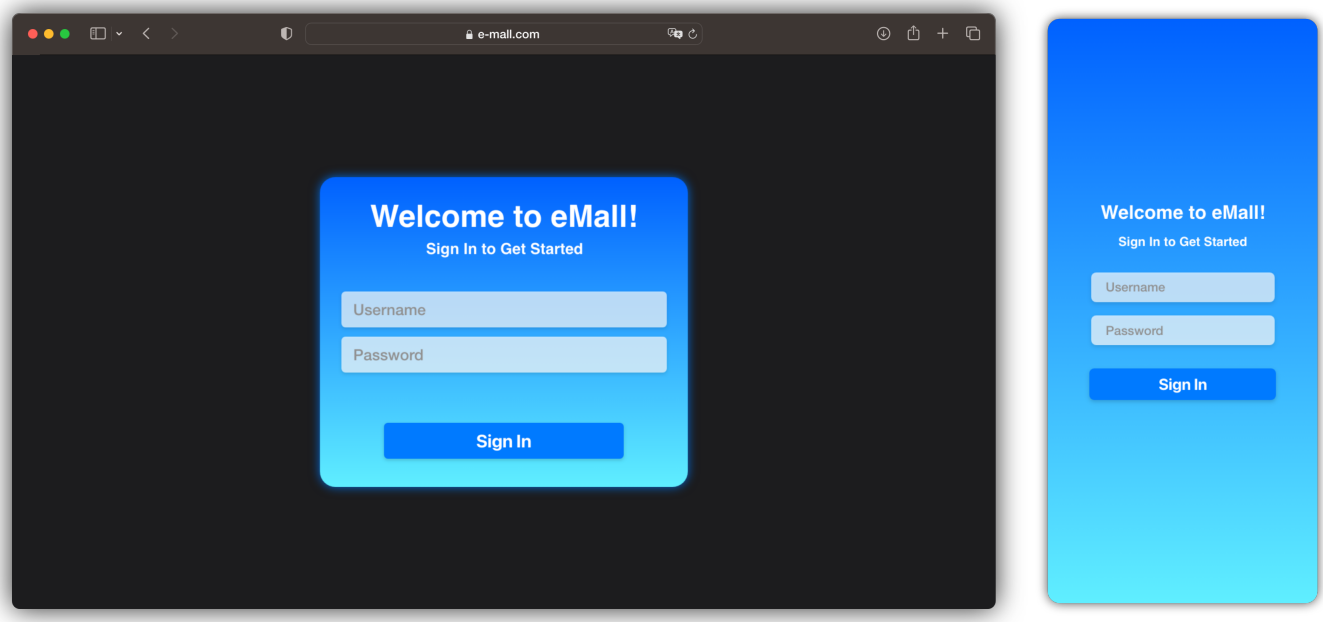

Figure 22: Login page mockup
Users will be able to log into their eMall account by inserting their credentials

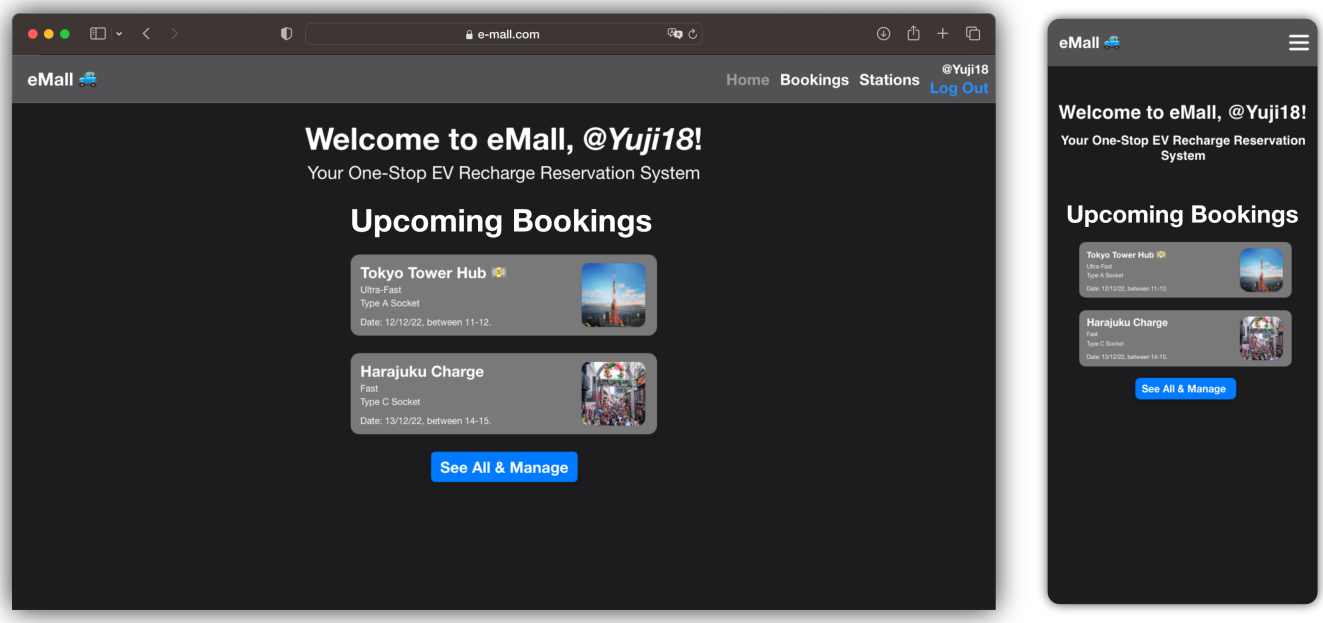

Figure 23: Home page mockup
Users will be able to click on the “See All” button to be redirected to the page with all their bookings

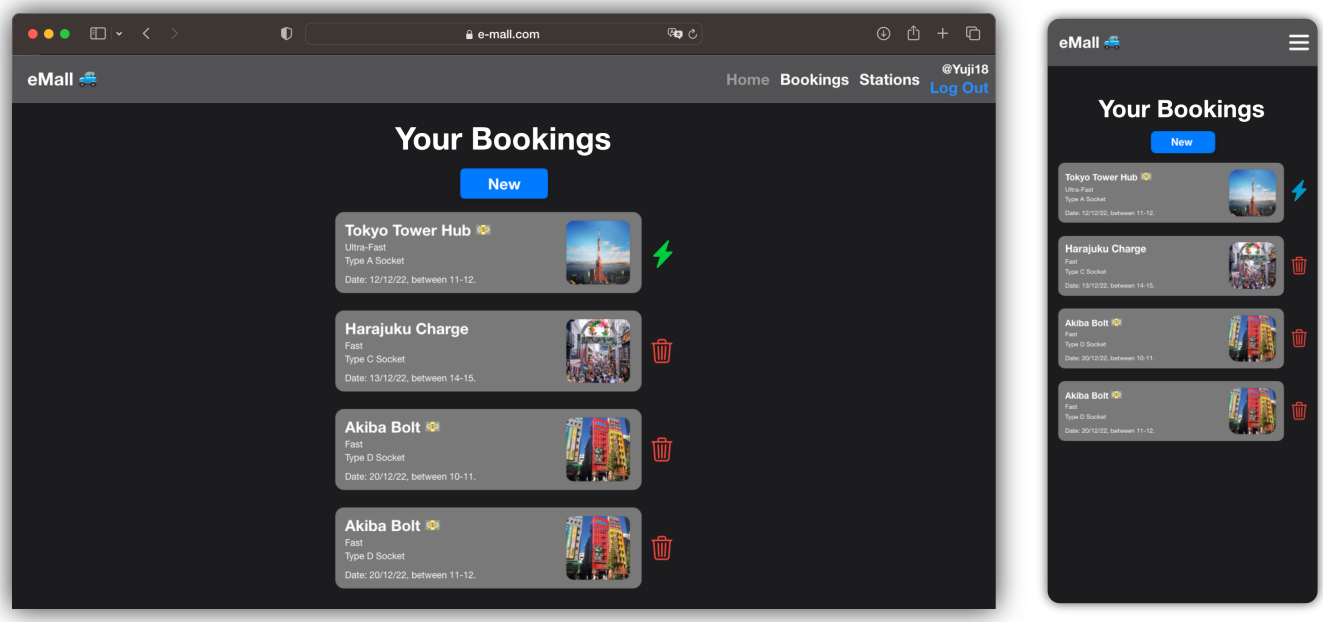

Figure 24: Bookings page mockup
Users will see the list of their bookings, with the option to delete any non-active one and to start their charge for any active booking with a vehicle connected

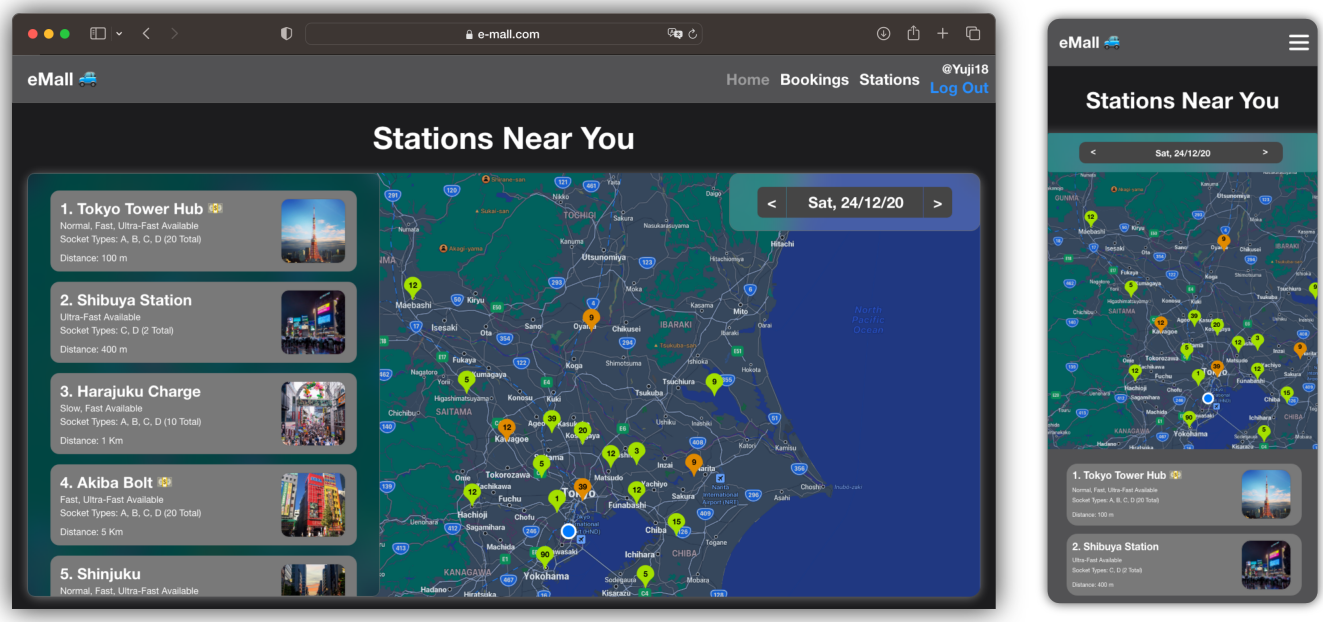

Figure 25: Stations page mockup
Users can use the map together with the time selector to filter the search

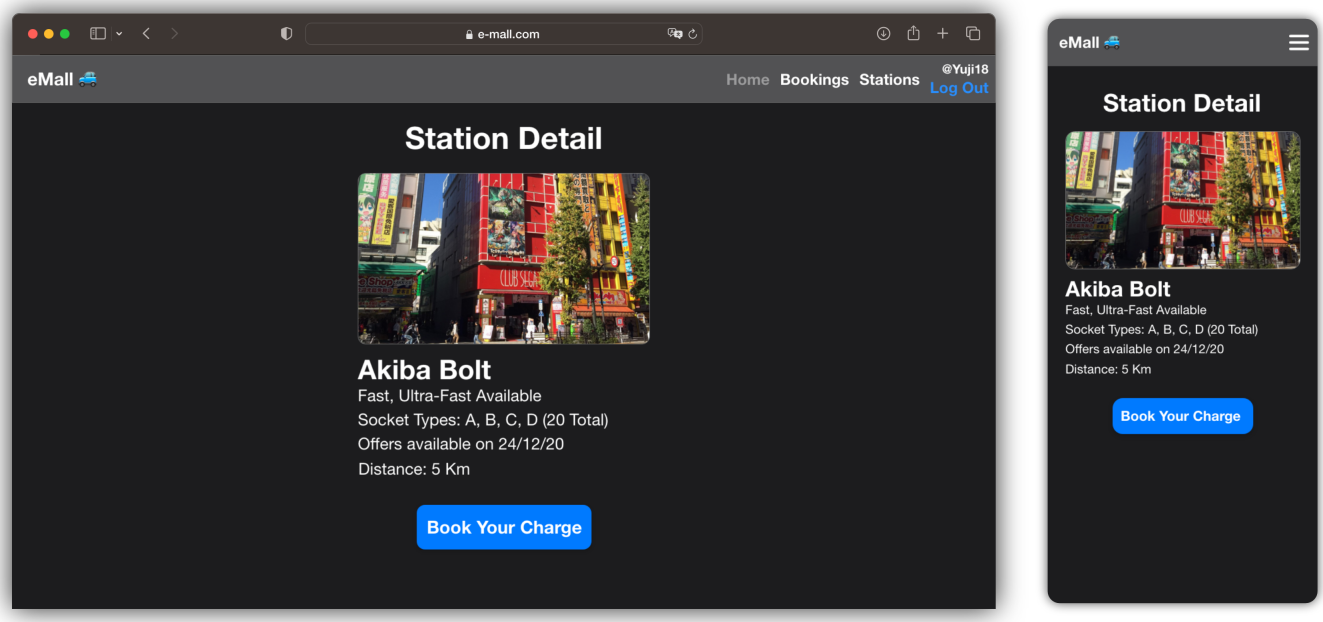

Figure 26: Station Details page mockup
Users can click on "Book Your Charge" to open the page for booking creation

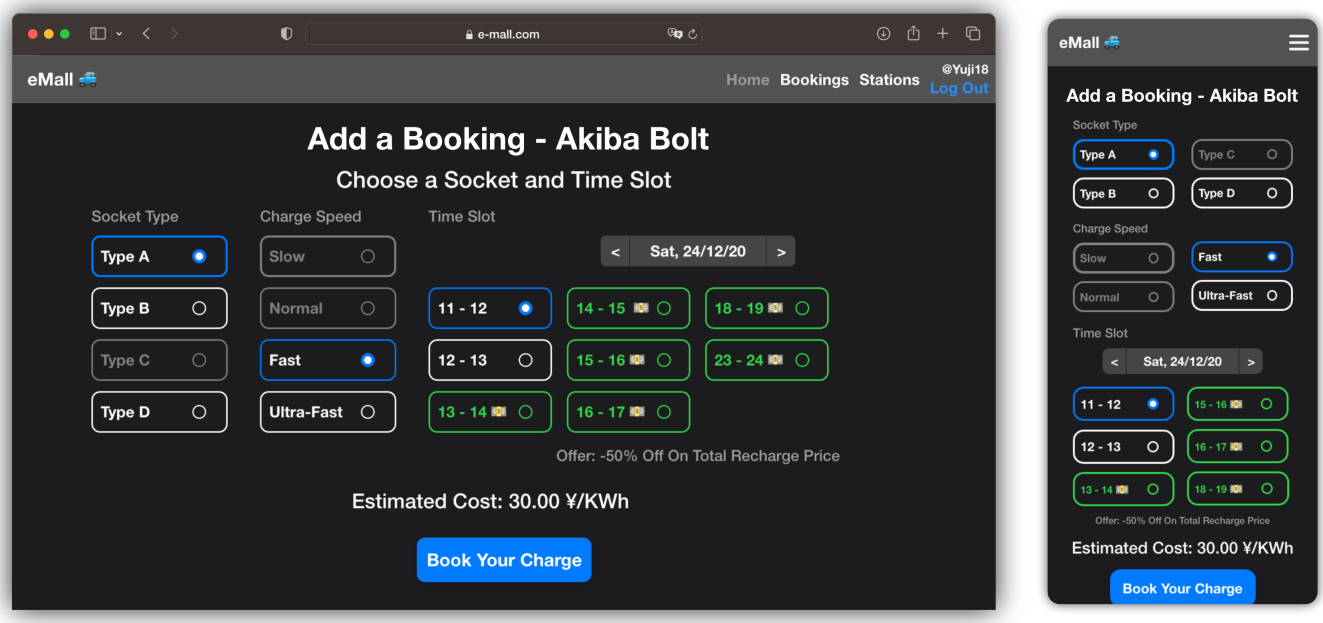

Figure 27: BookCharge page mockup
Users can select the time slot for their booking as well as charge speed and socket type among the available ones

Similarly, CPMSs will offer their functionality to CPOs via a web portal, here there is no need to support anything other than a desktop device, since CPMSs are intended to be managed by employees in a company environment. Follow some mockups for the CPMS's user interface.

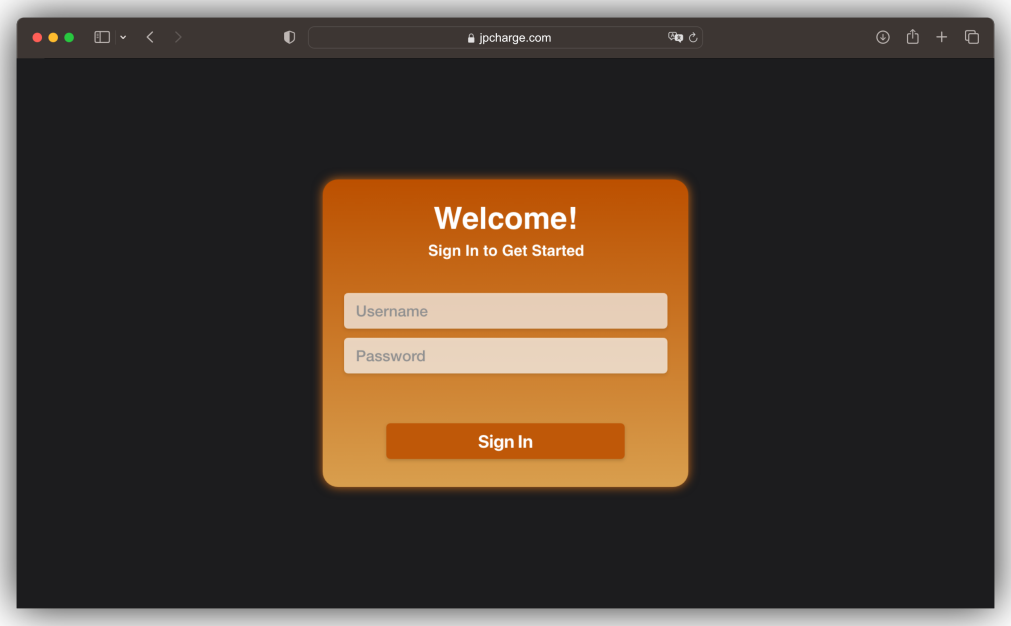

Figure 28: Signup page mockup
CPO Employees can input their company credentials to reach the CPMS management page

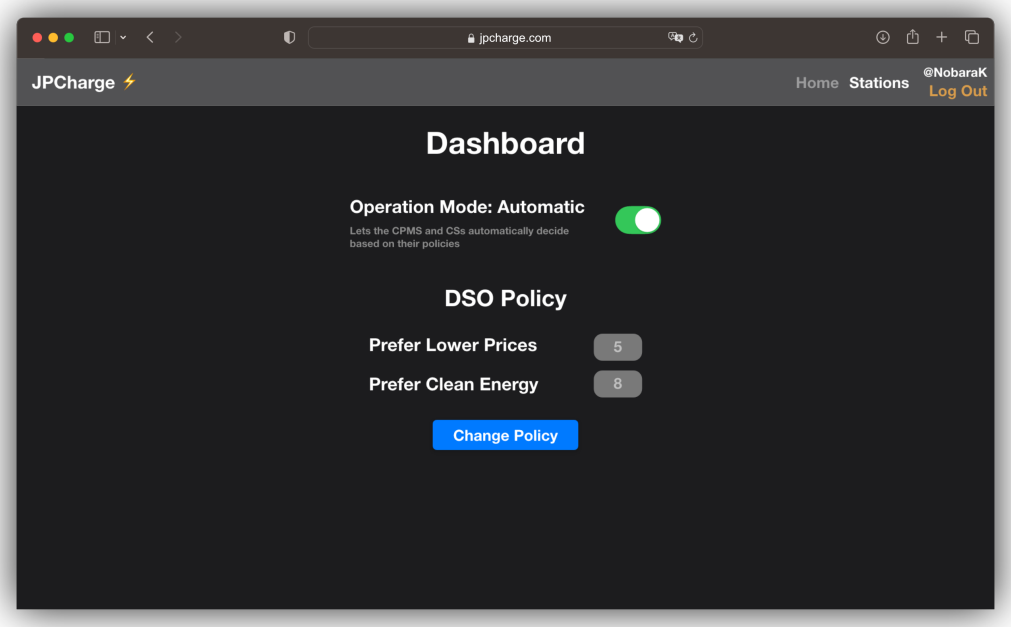

Figure 29: Main CPMS page
Here CPO Employees can toggle the CPMS automatic mode and set its policy for operating automatically

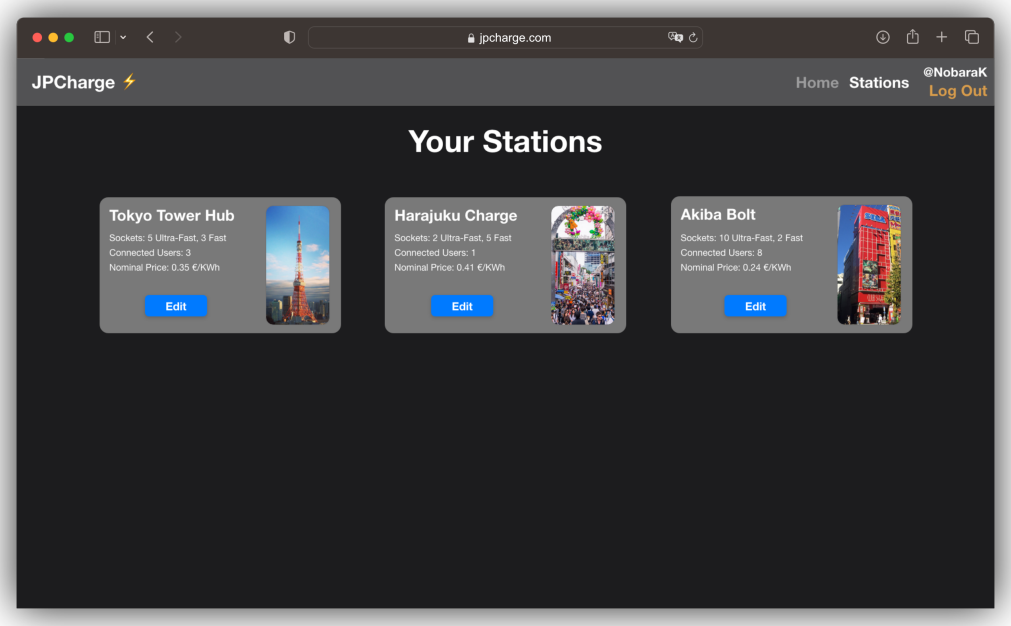

Figure 30: List of the CPMS's CS
CPO Employees see here the entire list of the CSs managed by their CPMS, and can choose to edit the details of any of them

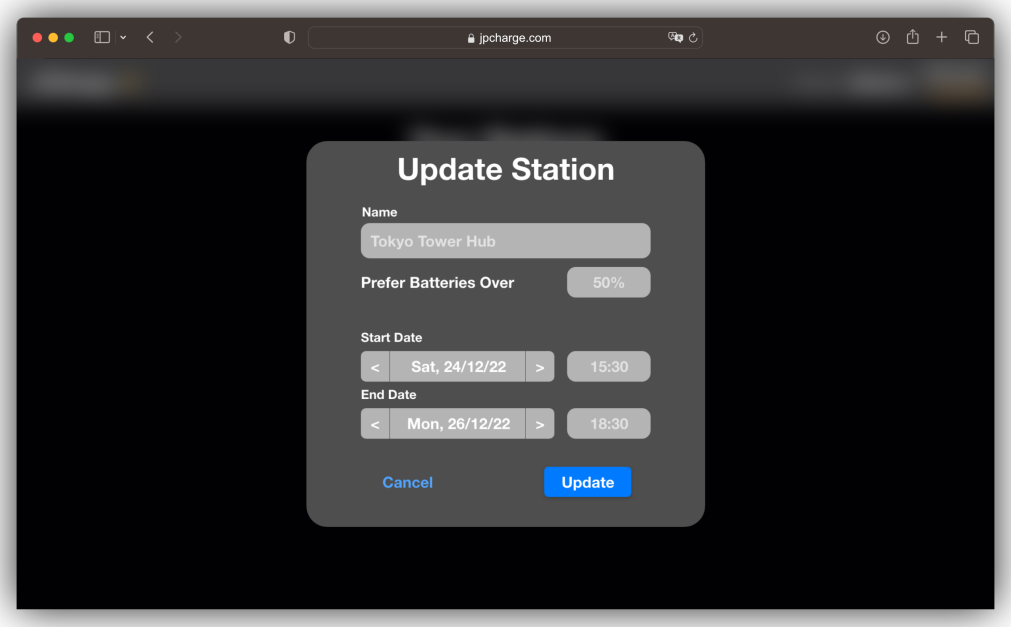

Figure 31: Details/Edit of a CS
CPO Employees see here the main properties of a CS and can alter them (for example, they can set the CS's prices, hence setting up offers

4 Requirements Traceability

The following table reports how the different components described in the architecture satisfy the requirements elicitated in the RASD document.
Note that two components will be omitted:

- The **Request Routers** of both the eMSP and CPMS, since they are implicitly required whenever both their back-end components and the front-end ones are present.
- The **DBMSs** are omitted since they are required whenever their respective Data Access components are needed.

Id	Requirement	Mappings
R1	The eMSP shall allow a registered user to login	eMSP: Web Application, Authentication, Login, Data Access CPMS: -
R2	The eMSP shall allow an unregistered user to register on the platform	eMSP: Web Application, Register, Payment Module, Data Access CPMS: -
R3	The eMSP should report to the user when they sent invalid input data through a form	eMSP: Web Application, Authentication, Login, Register, Search CS CPMS: -
R4	The eMSP should report to the user when they attempted to perform an unauthorized action	eMSP: Web Application, Authentication, Bookings Manager, Recharge Manager CPMS: -
R5	The eMSP should report to the user when the platform encountered an error while processing the action	eMSP: Web Application, Authentication, Login, Data Access CPMS: -
R6	The eMSP should report to the user any successful action performed	eMSP: Web Application, Authentication, Login, Register, Search CS, Bookings Manager, Recharge Manager CPMS: -
R7	The eMSP shall allow a logged-in user to view charging stations nearby	eMSP: Web Application, Authentication, Search CS, Data Access CPMS: CS List, Data Access
R8	The eMSP shall allow a logged-in user to know the cost of a recharge at a CS	eMSP: Web Application, Authentication, Search CS, Data Access CPMS: CS List, Data Access
R9	The eMSP shall allow a logged-in user to know about ongoing special offers present at a CS	eMSP: Web Application, Authentication, Search CS, Data Access CPMS: CS List, Data Access
R10	The eMSP shall allow a logged-in user to book a recharge at a free socket of a CS for a given time slot	eMSP: Web Application, Authentication, Search CS, Bookings Manager, Data Access CPMS: CS List, Data Access
R11	The eMSP shall allow a logged-in user to cancel any of its booked recharges before their scheduled time	eMSP: Web Application, Authentication, Bookings Manager, Data Access CPMS: -
R12	The eMSP shall not allow a user to book a recharge at a socket of a CS which has already been booked for the chosen time slot	eMSP: Web Application, Authentication, Bookings Manager, Data Access CPMS: -

Id	Requirement	Mappings
R13	The eMSP shall not allow a user to book multiple recharges with overlapping time slots	eMSP: Web Application, Authentication, Bookings Manager, Data Access CPMS: -
R14	The eMSP shall allow a logged-in user to see their bookings	eMSP: Web Application, Authentication, Bookings Manager, Data Access CPMS: -
R15	The eMSP shall allow a logged-in user to start the charging process for one of their booked recharges	eMSP: Web Application, Authentication, Bookings Manager, Recharge Manager, Data Access CPMS: Recharge Manager, CS Manager, Data Access
R16	The eMSP shall not allow a user without a booking to start the charging process	eMSP: Web Application, Authentication, Bookings Manager, Recharge Manager, Data Access CPMS: -
R17	The eMSP shall not allow a user to start the charging process as long as the socket is vacant	eMSP: Web Application, Authentication, Bookings Manager, Recharge Manager, Data Access CPMS: Recharge Manager, CS Manager, Data Access
R18	The eMSP should notify a logged-in user when their ongoing charging processes is complete	eMSP: Web Application, Authentication, Recharge Manager, Data Access CPMS: CS Manager, Data Access
R19	The eMSP should notify a logged-in user when their ongoing charging process is terminated due to the expiration of its booked time slot	eMSP: Web Application, Authentication, Recharge Manager, Data Access CPMS: CS Manager, Data Access
R20	The eMSP shall allow a logged-in user to terminate their charging process earlier	eMSP: Web Application, Authentication, Bookings Manager, Recharge Manager, Data Access CPMS: Recharge Manager, CS Manager, Data Access
R21	The eMSP should charge a user for everyone of his charging processes as soon as they finish	eMSP: Web Application, Authentication, Recharge Manager, Payment Module, Data Access CPMS: -
R22	The eMSP shall charge a user for a fee if they do not start a recharge during their booking's time slot	eMSP: Web Application, Authentication, Bookings Manger, Payment Module, Data Access CPMS: -

Id	Requirement	Mappings
R23	The eMSP shall be able to offer its functionalities to multiple users at once	eMSP: All components CPMS: -
R24	The CPMS should be able to acquire the location from CSs	eMSP: - CPMS: Web Application, Authenticate, CS List, Data Access
R25	The CPMS should be able to acquire the internal status from CSs	eMSP: - CPMS: CS Manager, Data Access
R26	The CPMS should be able to acquire the external status from CSs	eMSP: - CPMS: CS Manager, Data Access
R27	The CPMS should be able to start charging a vehicle connected to a CS socket	eMSP: - CPMS: CS Manager, Data Access
R28	The CPMS should be able to monitor the charging process of a vehicle	eMSP: - CPMS: CS Manager, Data Access
R29	The CPMS should be able to acquire from DSOs their current energy prices	eMSP: - CPMS: DSO Information
R30	The CPMS should be able to acquire from DSOs their energy mix	eMSP: - CPMS: DSO Information
R31	The CPMS shall allow its CPO to authenticate with it	eMSP: - CPMS: Web Application, Authenticate, Login, Data Access
R32	The CPMS shall allow its CPO to assign a DSO to a CS as its energy provider when operating in "Manual Mode"	eMSP: - CPMS: Web Application, Authenticate, CS List, CS DSO Manager, CS Manager, Data Access
R33	The CPMS should be able to automatically assign a DSO to a CS as its energy provider when operating in "Automatic Mode"	eMSP: - CPMS: CS DSO Manager, CS Manager, Data Access

Id	Requirement	Mappings
R34	The CPMS shall allow the CPO to decide the energy source management policies when operating in "Manual Mode"	eMSP: - CPMS: Web Application, Authenticate, CS List, CS Battery Policy Manager, CS Manager, Data Access
R35	The CPMS should be able to automatically choose the current energy source for a CS according to the given policy when operating in "Automatic Mode"	eMSP: - CPMS: CS Battery Policy Manager, CS Manager, Data Access
R36	The CPMS shall allow its CPO to assign a nominal-price, a user-price and an offer reset date to a CS when operating in "Manual Mode"	eMSP: - CPMS: Web Application, Authenticate, CS List, CS Prices Manager, CS Manager, Data Access
R37	The CPMS should be able to automatically assign a nominal-price, a user-price and an offer reset to a CS when operating in "Automatic Mode"	eMSP: - CPMS: CS Prices Manager, CS Manager, Data Access
R38	The CPMS shall allow the CPO to choose its policy for operating in "Automatic Mode"	eMSP: - CPMS: Web Application, Authenticate, Automatic Mode Manager
R39	The CPMS shall allow the CPO to choose whether it acts automatically or manually	eMSP: - CPMS: Web Application, Authenticate, Automatic Mode Manager
R40	The CPMS shall never take a decision automatically when it is set in "Manual Mode"	eMSP: - CPMS: Automatic Mode Manager, CS Battery Policy Manager, CS DOS Manager, CS Prices Manage

Id	Requirement	Mappings
R41	The CPMS should report to the user when they sent invalid input data	eMSP: - CPMS: Web Application, Authenticate, Login, Automatic Mode Manager, CS Battery Policy Manager, CS DOS Manager, CS Prices Manage, CS List
R42	The CPMS should report to the user when they attempted to perform an unauthorized action	eMSP: - CPMS: Web Application, Authenticate, Automatic Mode Manager, CS Battery Policy Manager, CS DOS Manager, CS Prices Manage, CS List, DSO Information
R43	The CPMS should report to the user when the platform encountered an error while processing the action	eMSP: - CPMS: Web Application, Authenticate, Login, Automatic Mode Manager, CS Battery Policy Manager, CS DOS Manager, CS Prices Manage, CS List, DSO Information
R44	The CPMS should be able to decide whether to start charging the internal batteries of a CS (if present)	eMSP: - CPMS: CS Battery Policy Manager, CS Manager, Data Access

5 Implementation, Integration and Test Plan

Here we discuss how the architecture previously described will be realized, with its modules integrated and tested in order to guarantee the correctness of the final product, w.r.t. the requirements and directions expressed both in this document and the RASD, during each step of development.

5.1 Implementation

The implementation will proceed separately for the two main parts that the system is divided into, the eMSP and the CPMS, since the two can easily be implemented just by knowing the exposed interfaces of the other and no further knowledge. That being said, withing each part the implementation order can be the following:

5.1.1 eMSP

1. The **Database** is the first component to implement as the way data is organized within it has to be reflected by the Data Access component.
2. The **Data Access** component can be realized immediately after the DB, as it allows access from the other modules to the data it stores and is also relatively simple, mainly realizing ORM.
3. The **Payment Module** as it is the gateway to interfaces with the Payment Provider, it needs to be implemented before the components which need access to the former.
4. The following components can all be implemented concurrently as they have no dependencies with one-another and all constitute the interfaces which the Web Application relies upon.
    - Login
    - Register
    - Serch CS
    - Bookings Manager
    - Recharge Manager
5. The **Authentication** component can be realized after the other back-end components, since all of them either rely on authenticated connections or grant them to clients, but only the request router uses this component to verify the session of a connected client.
6. The **Web Application** is realized for last as it is the less critical component and its testing can be highly aided by the existence of all the other back-end components. To be noted that the Web Application's UI design, not its functionalities, can be realized at any time, while the development of the application is bonded to this sequence.

5.1.2 CPMS

1. The **Database** is the first component to implement as the way data is organized within it has to be reflected by the Data Access component.
2. The **Data Access** component can be realized immediately after the DB, as it allows access from the other modules to the data it stores and is also relatively simple, mainly realizing ORM.
3. The **CS Manager** is a critical component that needs to be realized before the others as it handles communications with the CSs and offers the functionalities that allow other components to configure CSs.

4. The following components can all be implemented concurrently as they have no dependencies with one-another, they either expose and API for the CPO's web application or an API intended for the eMSP.
    - Login
    - CS Battery Policy Manager
    - CS DSO Manager
    - CS Prices Manager
    - Automatic Mode Manager
    - CS List
    - Recharge Manager
    - DSO Information
5. The **Authentication** component can be realized after the other back-end components, since all of them either rely on authenticated connections or grant them to clients, but only the request router uses this component to verify the session of a connected client.
6. The **Web Application** is realized for last as it is the less critical component and its testing can be highly aided by the existence of all the other back-end components. To be noted that the Web Application's UI design, not its functionalities, can be realized at any time, while the development of the application is bonded to this sequence.

5.2 Integration and Testing

In this section we present our integration and testing plan for both systems. Since the two systems are designed to be distinct, they are first tested separately, following two parallel bottom-up approaches while stubbing the CPMS (for the eMSP) and writing a driver of the eMSP (for the CPMS). After all tests have been carried out (that is, all unit tests and bottom-up integration testing) the final integration of the two systems is carried out by removing stubs and drivers and using the pre-defined interfaces for communication between the two parties.
As hinted above, the two systems will use the following testing plan:

1. **Unit Testing** of each component;
2. **Bottom-Up Integration Testing** with appropriate drivers;
3. **Final Integration Testing** by removing the CPMS stub (on the eMSP side) and the eMSP-like driver (on the CPMS side)

This plan allows for easier bug identification and fixing, and ensures that whenever a component is finished its behavior is fully compliant with all the requirements.
It should be noted, however, that some components must be assumed to work as expected, since they are external modules that cannot be tested. An example of this is the DBMS, which is stubbed in all unit tests.
In addition, a variety of System Tests are used to ensure that properties of the system as a whole are respected. In particular, the following tests are adopted:

- **Functional Testing**, to make sure that each requirement specified in the RASD document has been implemented properly;

- **Performance Testing**, to analyze key metrics of the performance of the system (such as response time) and identify and remove possible bottlenecks that would hamper the user experience;
- **Stress Testing**, to ensure that the system remains stable even under heavy or unforeseen load conditions.

Below is a list of diagrams illustrating the bottom-up integration testing approach and how all modules are progressively integrated and tested together.

5.2.1 eMSP Integration

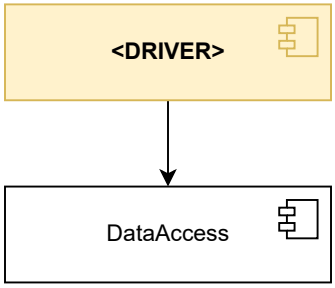

Figure 32: Data Access component integration

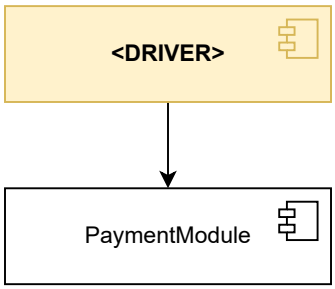

Figure 33: Payment Module component integration

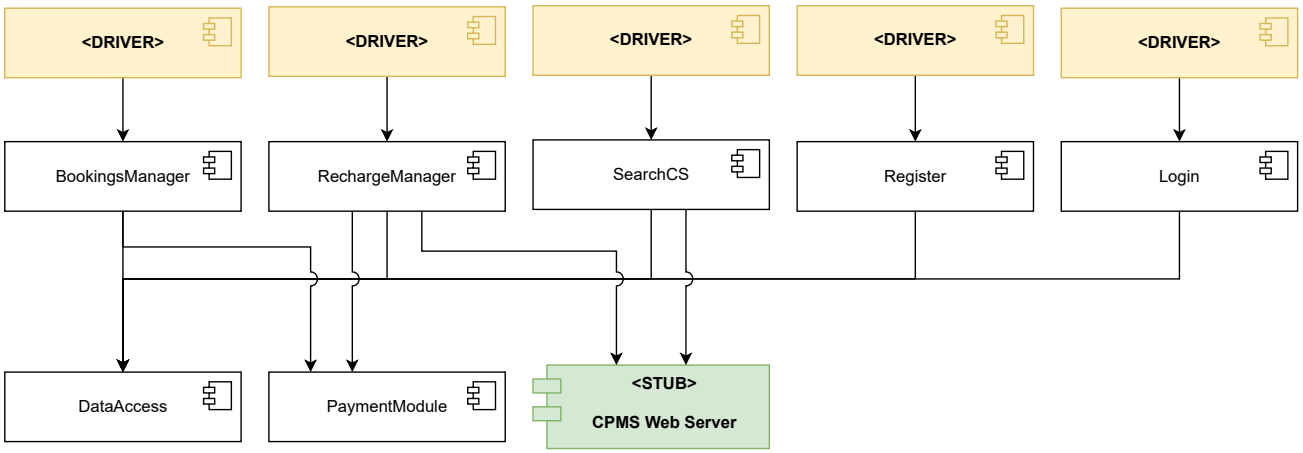

Figure 34: Bookings Manager, Recharge Manager, SearchCS, Register and Login components integration

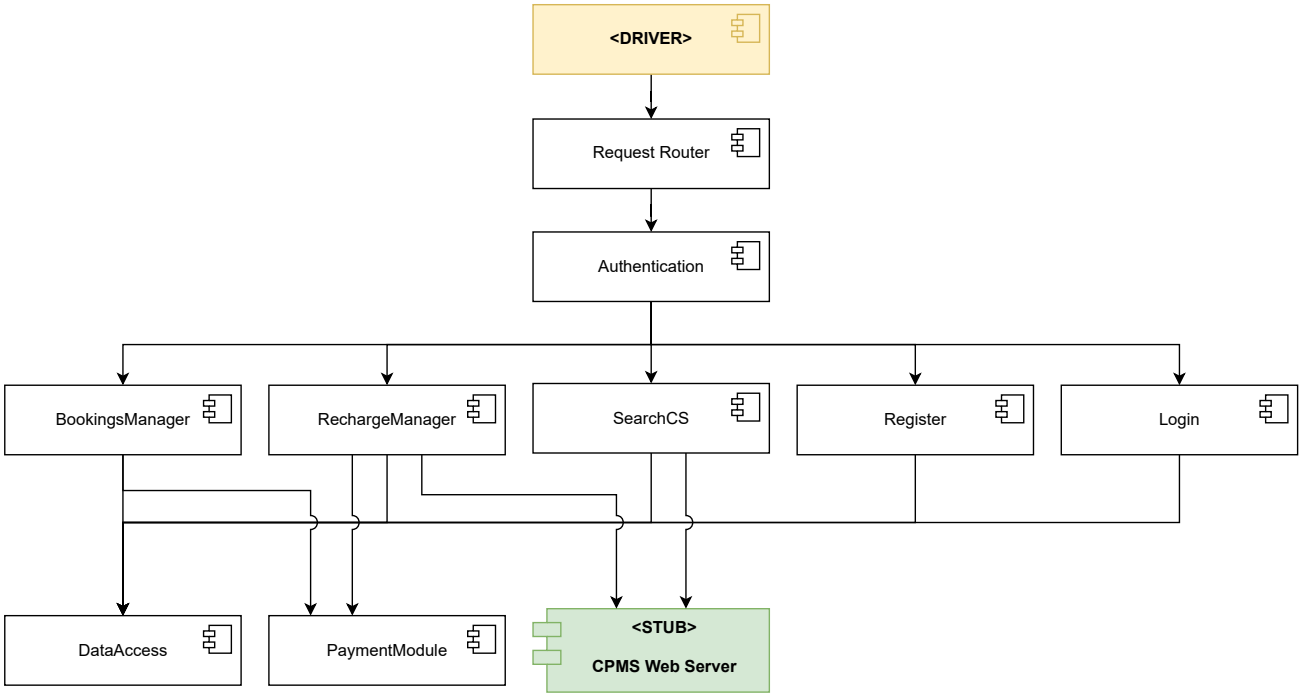

Figure 35: Authentication and Request Router components integration

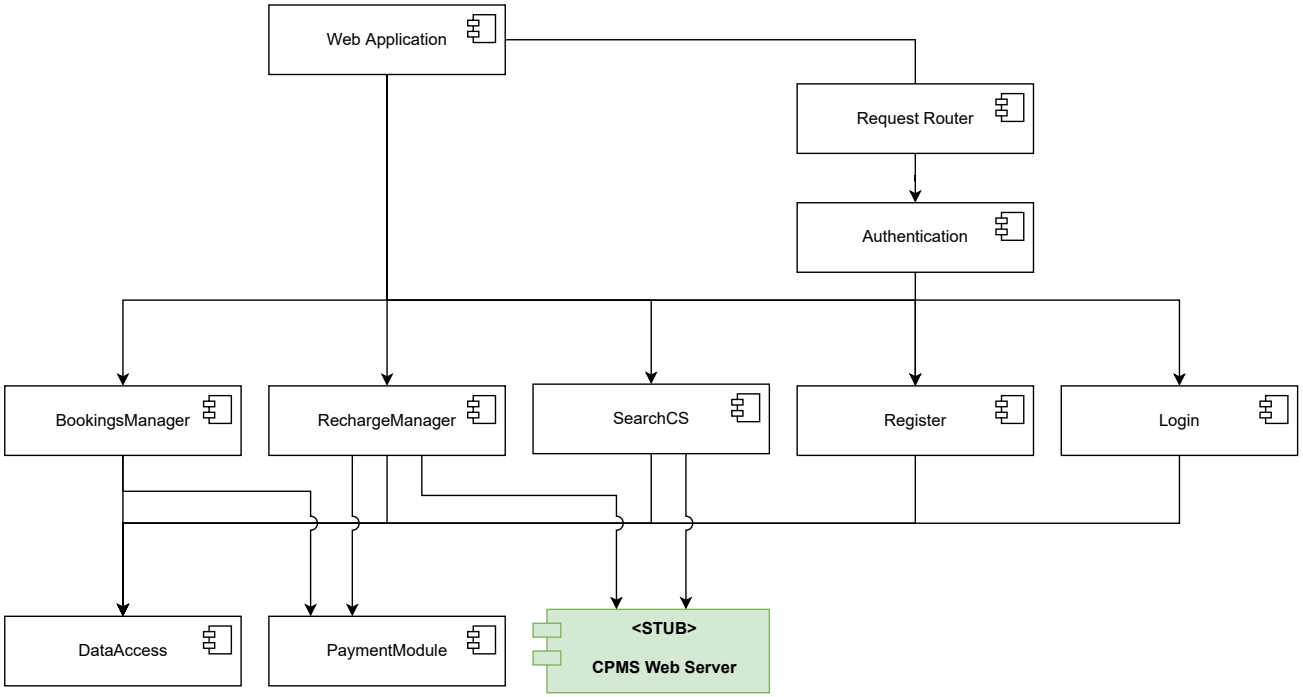

Figure 36: Full integration

5.2.2 CPMS Integration

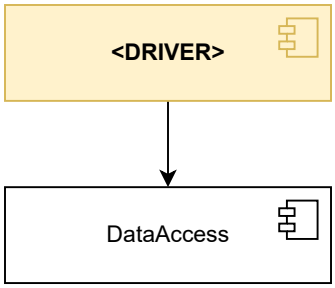

Figure 37: Data Access component integration

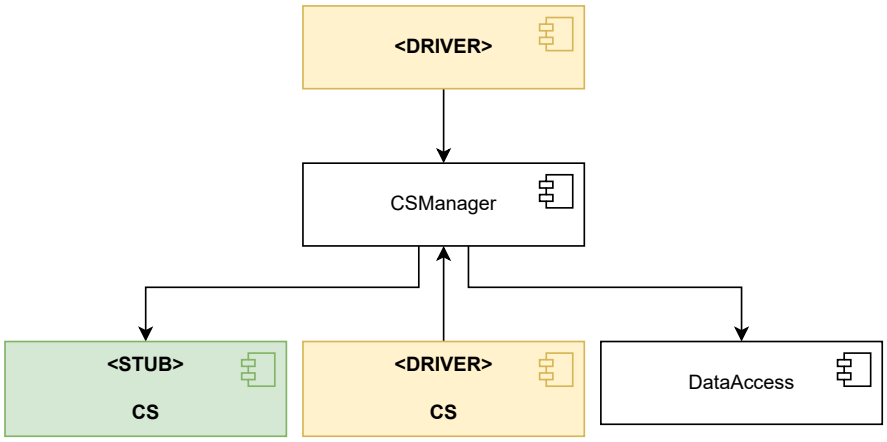

Figure 38: CS Manager component integration

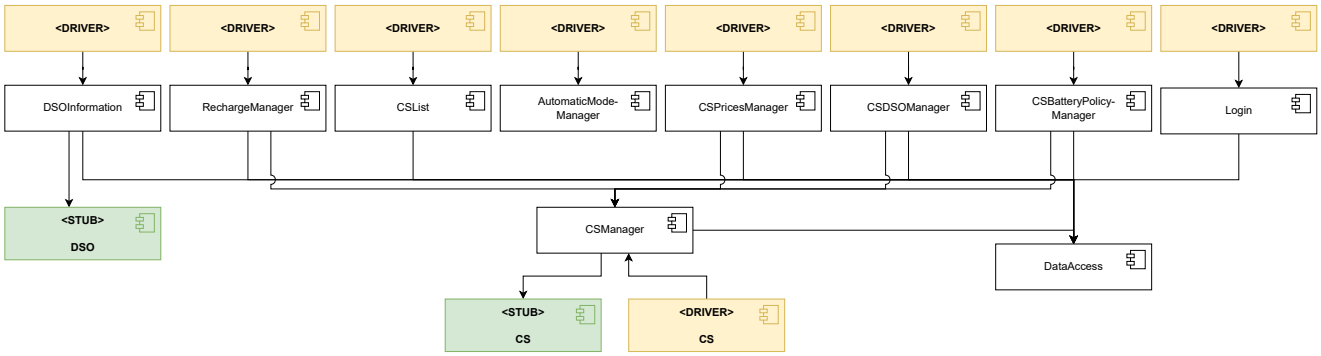

Figure 39: DSO Information, Recharge Manger, CS List, Automatic Mode Manager, CS Prices Manager, CS DSO Manager, CS Battery Policy Manager and Login component integration

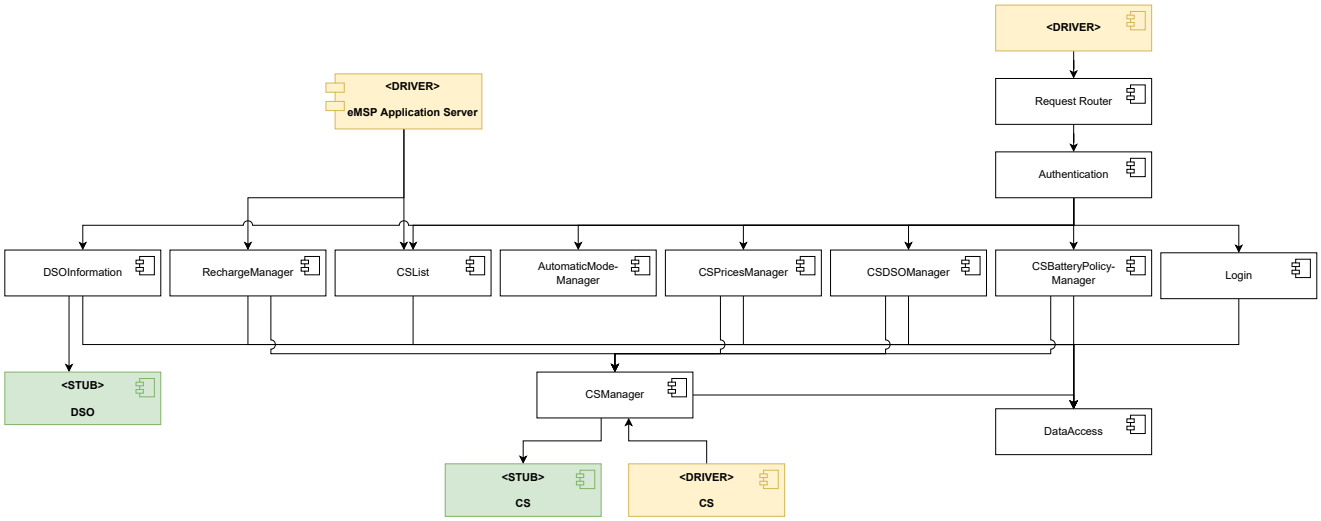

Figure 40: Authentication and Request Router component integration

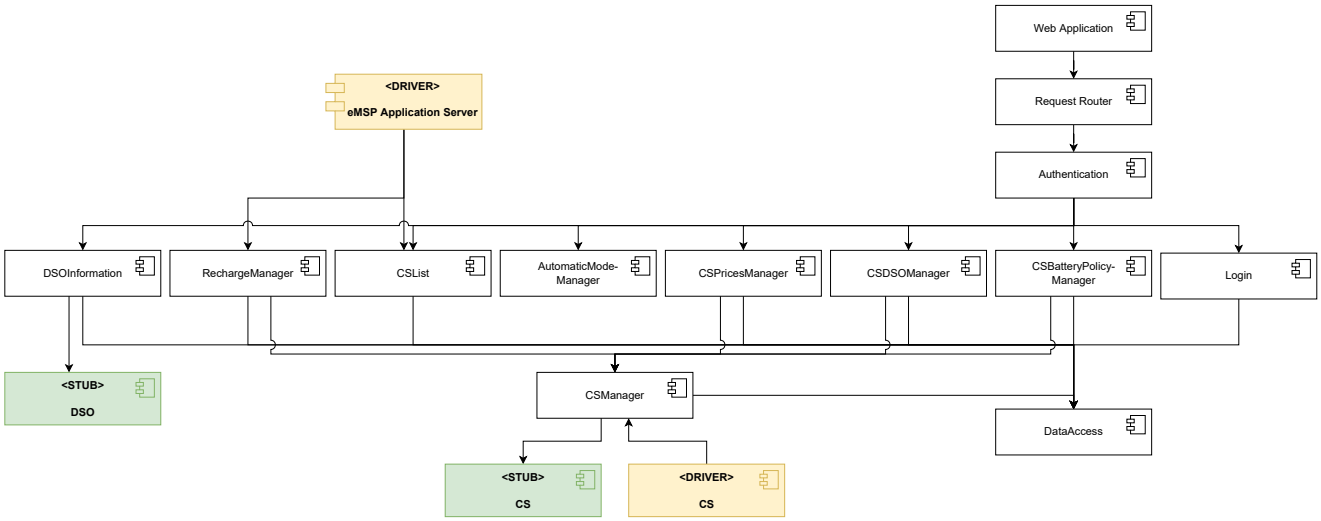

Figure 41: Full integration

5.2.3 Whole System Integration

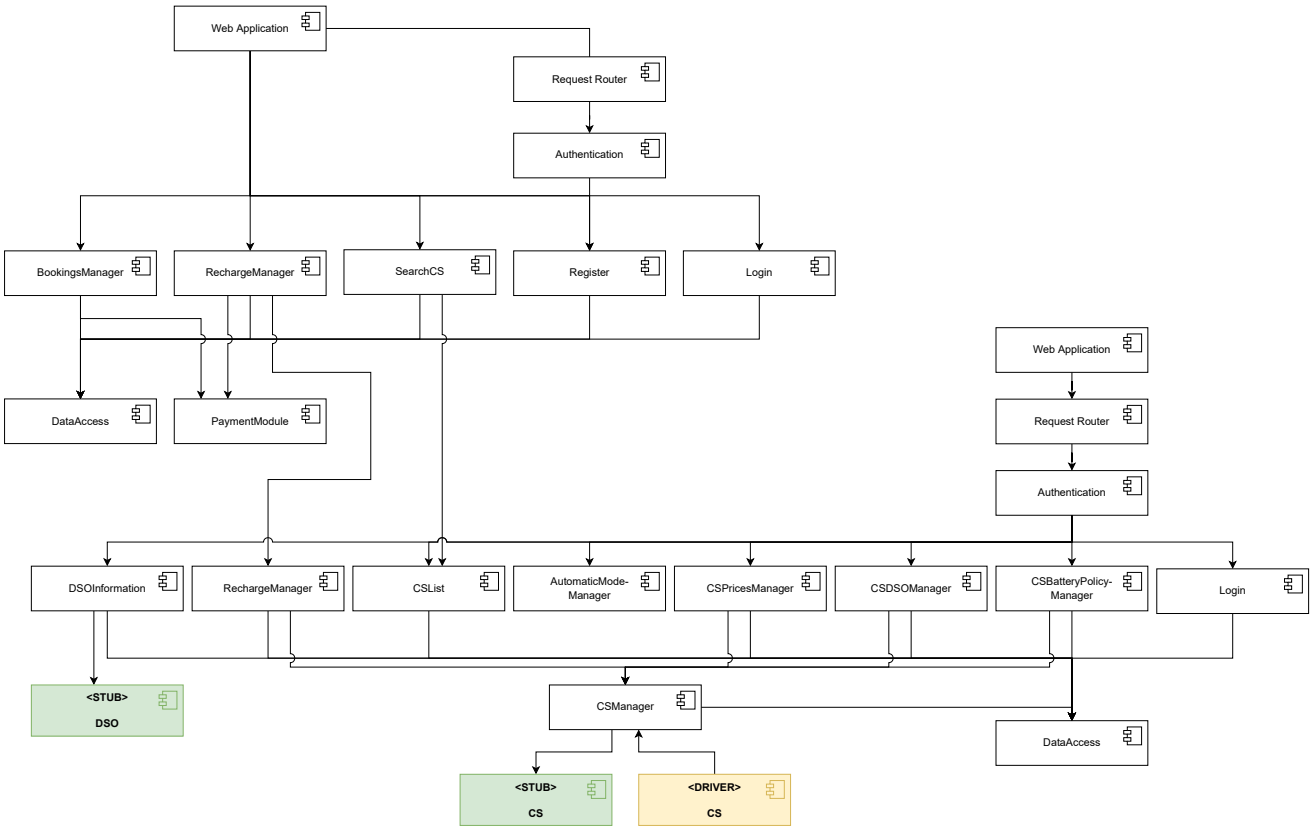

Figure 42: eMSP and CPMS integration

6 Effort Spent

6.1 Ronzani Marco - mat: 224578

Task	**Time spent**
Introduction	∞h
Architectural Design	∞h
User Interface Design	∞h
Requirements Traceability	∞h
Implementation, Integration and Test Plan	∞h
Other	∞h
Total	$5 * \infty h$

6.2 Sassi Alessandro - mat: ...

Task	**Time spent**
Introduction	∞h
Architectural Design	∞h
User Interface Design	∞h
Requirements Traceability	∞h
Implementation, Integration and Test Plan	∞h
Other	∞h
Total	$5 * \infty h$

7 References

1. ChargeLab - operating system for EV charges
2. Platform for Electromobility. EV Charging: How to tap in the grid smartly?.
3. F. Campos, L. Marques, and K. Kotsalos, Electric Vehicle CPMS and Secondary Substation Management.
4. Shu Su, Hui Yan, and Ning Ding. 2018. Machine Learning-Based Charging Network Operation Service Platform Reservation Charging Service System. In Proceedings of the 2018 International Conference on Signal Processing and Machine Learning (SPML '18). Association for Computing Machinery, New York, NY, USA, 1–5.
5. Jan Mrkos, Antonín Komenda, and Michal Jakob. 2018. Revenue Maximization for Electric Vehicle Charging Service Providers Using Sequential Dynamic Pricing. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 832–840.