



POILTECNICO

MILANO 1863

e-MALL
e-Mobility for All

DD
Design Document

Sara Limooee, 10886949
Parham Ebadi, 10870289

Fall 2022

Table of Contents

1. INTRODUCTION	1
1.1. PURPOSE.....	2
1.2. SCOPE.....	2
1.3. DEFINITIONS, ACRONYMS AND ABBREVIATIONS	2
1.3.1. Definitions	3
1.3.2. Acronyms	3
1.3.3. Abbreviations.....	3
1.4. REVISION HISTORY	2
1.5. REFERENCE DOCUMENTS.....	2
1.6. DOCUMENT STRUCTURE	2
2. ARCHITECTURAL DESIGN	1
2.1. OVERVIEW: HIGH LEVEL COMPONENTS AND THEIR INTERACTION	2
2.2. COMPONENT VIEW	2
2.3. DEPLOYMENT VIEW.....	2
2.4. RUNTIME VIEW	2
2.4.1. General Request with Authorization	3
2.4.2. EV-Driver Registering	3
2.4.3. Login.....	3
2.4.4. Searching for Charing Stations	3
2.4.5. Booking.....	3
2.4.6. Payment	3
2.4.7. Rating	3
2.4.8. CPO View Internal/External Status	3
2.4.9. CPO View Stations Locations	3
2.4.10. CPO View Stations Locations	3
2.5. COMPONENT INTERFACES	2
2.6. SELECTED ARCHITECTURAL STYLES AND PATTERNS.....	2
2.7. OTHER DESIGN DECISIONS.....	2
3. USER INTERFACE DESIGN	1
3.1. EV-DRIVER	2
3.1.1. Register	3
3.1.2. LOGIN.....	2
3.1.3. Searching for Charing Stations	3
3.1.4. Booking.....	3
3.1.5. Payment	3
3.1.6. Rating	3
3.2. CPO	2
3.2.1. Login.....	3
3.2.2. View Stations' Locations	3
3.2.3. View Internal Status	3
3.2.4. View External Status	3
3.2.5. View Battery Status	3
4. REQUIREMENT TRACEABILITY.....	1
5. IMPLEMENTATION, INTEGRATION AND TEST PLAN	1
3.1. IMPLEMENTATION	2
3.1. INTEGRATION & TEST PLAN	2
6. EFFORT SPENT	1
7. REFERENCES	1

1. Introduction

1.1. Purpose

In recent years, due to world warming and the increase in the number of pollution in the air produced by fuel vehicles because of using petrol, gasoline etc., there is a huge need for models of cars which must be less disastrous for the environment. Hybrid electric vehicles (HEVs) and electric vehicles (EVs) use less or even no fuels so they have much less effect on the environment. However, these kinds of cars need to be charged whenever their battery is low. The idea is to develop an electric Mobility Service Provider(eMSP) so that users who are EV drivers can easily make a decision among various charging points and book a place for charging their electric vehicles based on some factors e.g., distance, price, etc. The goal of this document is to provide more technical information about the eMSP application and its interaction with different components of the eMSP system and the CPMS system of charging stations. In fact, developers of the application use this document as a guide to developing the application. In general, the main different features of this document are:

- The high-level architecture
- Main components of the system
- Interfaces provided by the components
- Design patterns adopted
- Implementation, integration, and testing

1.2. Scope

In order to help EV-drivers to find nearby charging stations for their electric car and know about any special offer that either the eMSP or the CPMS (means any offer or suggestions that the CPOs give to the customers), The application has the following parts:

1. EV-driver login to the eMSP system and inserts data about his/her car. According to this information, he can search for charging stations by various filters e.g., distance, price, type of sockets, etc. He can also pay for the obtained service through the interface provided by the eMSP.
2. CPOs log in to the CPMS system to control the general internal and external status of their charging stations, the current price of energy obtained from each DSO that the charging station is working with and to control some other functionalities that can be also controlled by humans manually.
3. eMSP system interacts with the CPMS system to start the charging process, shows the charging process to the end user (EV-driver), giving information about the available charging stations and free sockets of any type of socket (slow, fast, rapid) and their prices and generally any information that must be changed between eMSP and CPMS.

EV-drivers might receive some special offers either from eMSP system for using the application or from CPMSs. Moreover, some suggestions can be given to EV-drivers about when to charge their car based on the previous charging and the distance that the car has passed (this information can be obtained by an API between the CPMS and the navigation system of the car in order to estimate how much distance a car can go at most). Since this project has been done by a group of two students, the last point mentioned above, the suggestions given by CPMS to the EV-drivers have not been considered in the project.

1.3. Definitions, Acronyms and Abbreviations

1.3.1. Definitions

Definitions	Descriptions
External Status	number of charging sockets available, their type, cost, and estimated amount of time until the first occupied socket is freed
Internal Status	amount of energy available in its batteries, number of vehicles being charged, amount of power absorbed, and remaining time of the charge of each vehicle
Notification	A message shown to the user by the system when he/she must be notified about something (ex: when a new offer is available, or when the charging process starts or finishes)
API	Stands for Application Programming Interface. APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

1.3.2. Acronyms

Acronyms	Descriptions
eMPS	e-Mobility Service Provider
DD	Design Document
CPMS	Charge Point Management System
DSO	Distribution System Operators
API	Application Programming Interface

1.3.3. Abbreviations

Abbreviations	Descriptions
G	Goal
R	Requirement
C	Component

1.4. Revision History

Version	Date	Modifications
1.0		
2.0		

1.5. Reference Documents

- Specification Document: “Assignment RDD AY 2022-2023_v2.pdf”
- Course slides

1.6. Document Structure

- Section 1
In this section a brief description about design document is given and the purpose and the scope are introduced. This section also includes definitions, acronyms and abbreviations.
- Section 2
This part is the fundamental part of design document which includes the component diagram and interfaces, deployment view, runtime view and architectural patterns chosen with the other design decisions.
- Section 3
This section includes the user interface mockups with more details and more completely than the mockups in RASD document.
- Section 4
This section contains the traceability matrix that shows which components satisfy certain requirements.
- Section 5
This section includes the implementation plan, integration plan and testing plan, and also shows how these plans are executed.
- Section 6
The amount of time spent for each section and each member of the group is included in this section.

2. Architectural Design

2.1. Overview: High-level Components and their Interactions

In this project, as discussed in the RASD document, there are two individual systems in the eMALL, eMSP and CPMS systems. Each of these systems follows the three-tier pattern with presentation layer, business logic layer and a data layer(tier)¹.

- Presentation tier

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for example. Web presentation tiers are usually developed using HTML, CSS and JavaScript. Desktop applications can be written in a variety of languages depending on the platform.

- Application tier

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

The application tier is typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using API calls.

- Data tier

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB or MongoDB.

\$\$\$\$\$\$\$\$\$
\$\$\$\$\$\$\$\$\$
\$\$\$\$\$\$\$\$\$
\$\$\$\$\$\$\$\$\$
Picture done

2.2. Component View

\$\$\$\$\$\$\$\$\$
\$\$\$\$\$\$\$\$\$
\$\$\$\$\$\$\$\$\$
What is the difference between provided and required?
Image result for provided interface and required interface

¹ Layer and tier words can be used interchangeably.

Required and provided interfaces appear to be UML-related terms, where a provided interface describes functionality offered by a class and required interfaces describe functionality needed by another class: further reading. In Java, all interfaces are the same; there is no distinction between provided/required.

Picture done

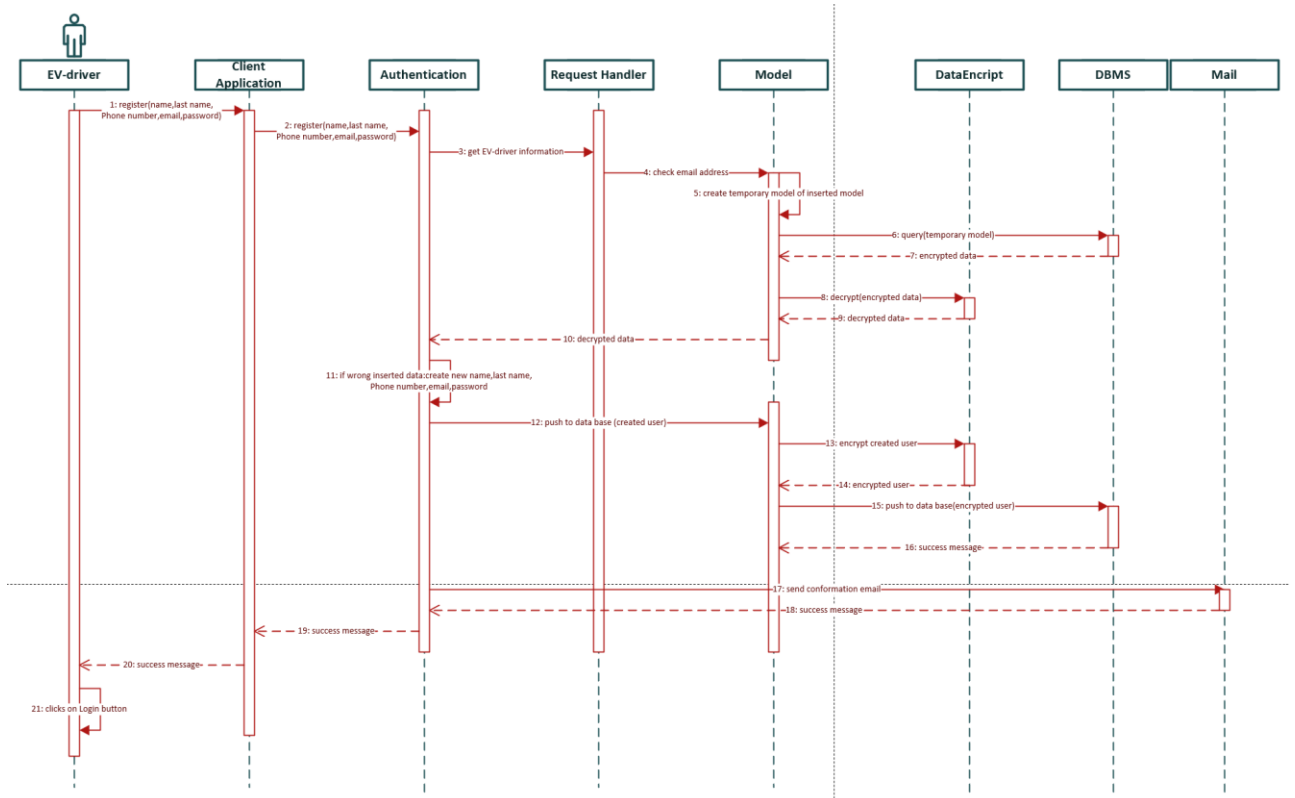
2.3. Deployment View

2.4. Runtime View

In the following sequence diagrams, we discuss the detailed interactions between user and each component. The main interactions between user and system have been already discussed in RASD. In order to keep the diagrams readable, only the main flow of the events is visualized and exceptions are not considered.

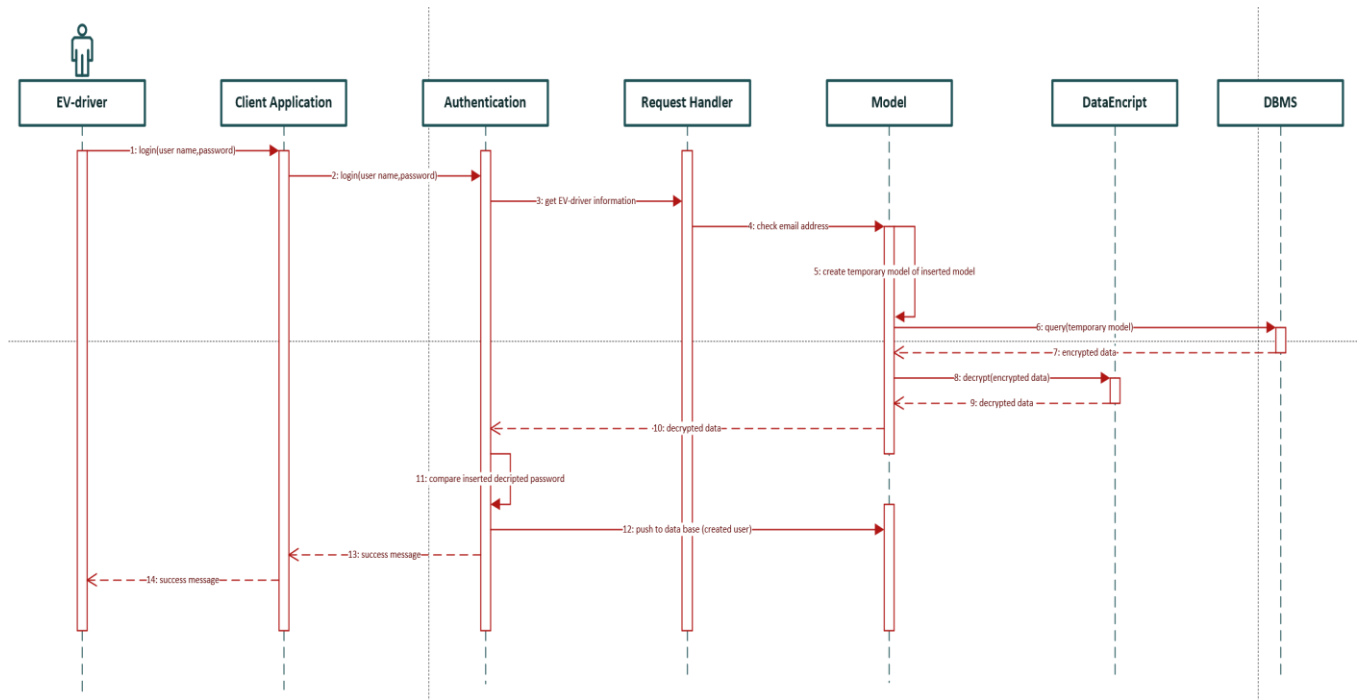
2.4.1. EV-Driver Register

EV-driver enters personal information (phone number, email, first name and last name, password) through Client application component (EMSP). Then the information is sent to Authentication component. Now, it is time to check if another EV-driver with same phone number or email address have already signed up in the application or not. This is checked in Model component. So, the Authentication component sends the information to Model component. Model sends a query to DBMS. The answer of DBMS is encrypted for security purposes. So, there is a DataEncrypt component which decrypts the reply from DBMS. If the retrieved data is null, it means there is no user with entered phone number and email address. Then the final step is to send an email through MailService component to complete the registration. When the user clicks on the confirmation link, he/she will be redirected to a web page. Finally, a query will be sent to database to update this user flag to True.



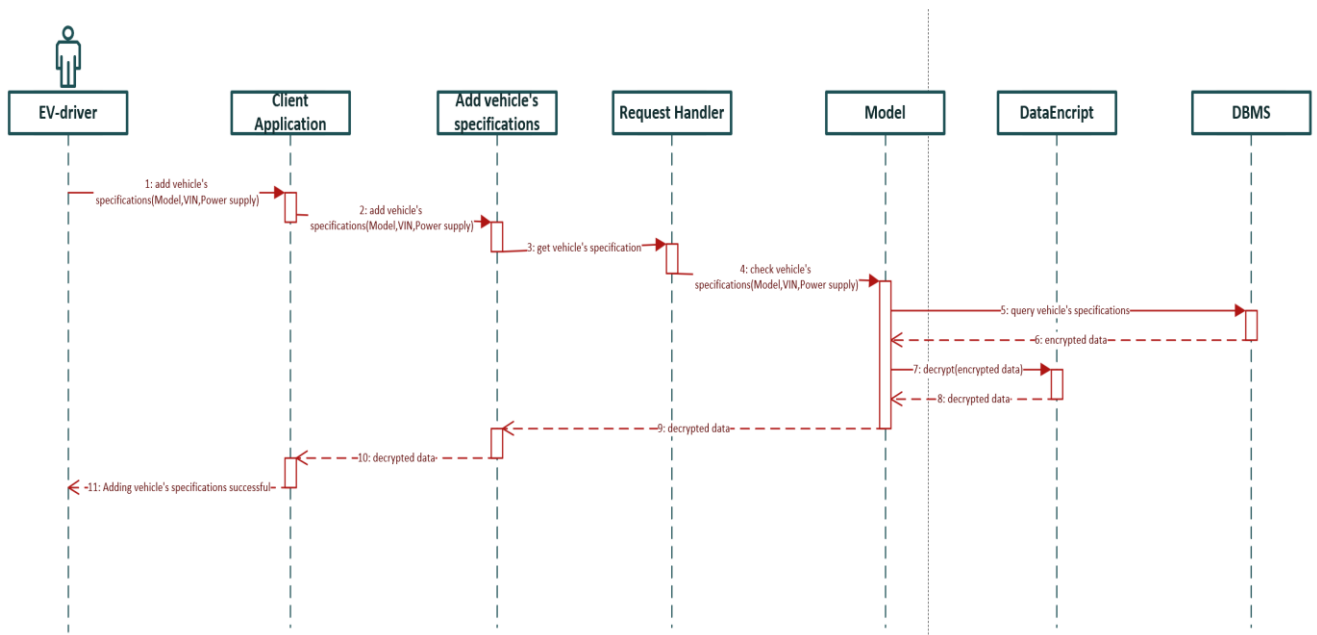
2.4.2. Login EV-driver

EV-driver enters email and password through EMSP component. Then, his/her information is sent to Authentication component. Now, it should be checked if the user has registered before or not. This is checked in Model component. So, the Authentication component sends the information to Model component. Model sends a query to DBMS to check if the email has registered in the app before or not. The answer of DBMS is encrypted for security purposes. So, there is a DataEncrypt component which decrypts the reply from DBMS. If the retrieved data is null, it means there is no user with the entered email. Otherwise, the user exists. Now, the Authentication component should check if the entered password is the same as decrypted password retrieved from DBMS or not. If the check result is positive, the retrieved user is returned, otherwise a null value is returned.



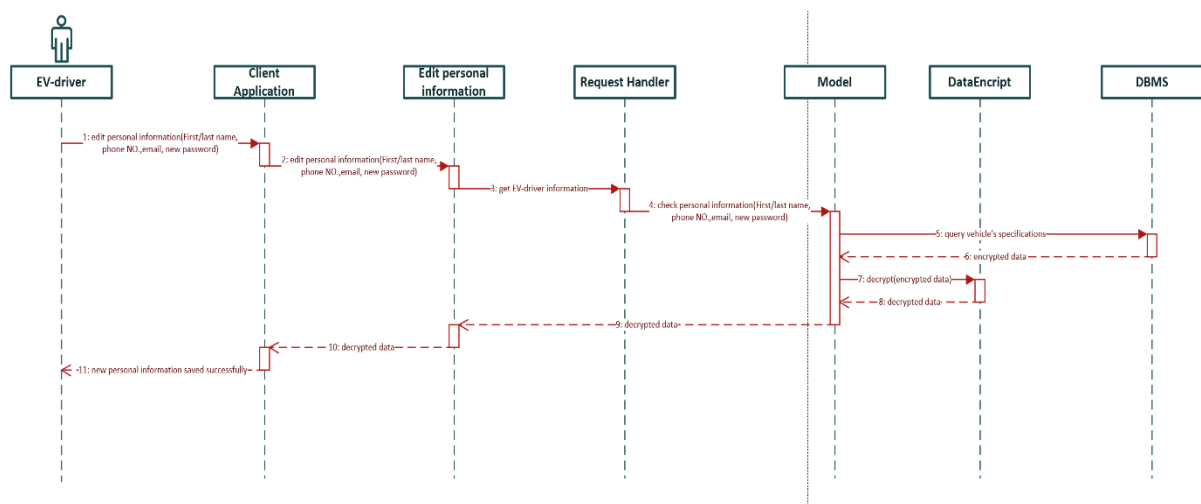
2.4.3. Add vehicle's specifications

EV-driver enters vehicle's specifications in the client application (EMSP) component. The request is then sent to request handler component to check to which component should the request be sent for being handled. Then the request and vehicle specifications are sent to Modify Information component. Then this component, sends the information to Model. Model sends a query to database to update the vehicle specifications of this user. Finally, DBMS sends an encrypted result which should be first decrypted through DataEncrypt component. If the retrieved data from DataEncrypt component is not null, it means that the new vehicles specifications has been added to DBMS successfully.



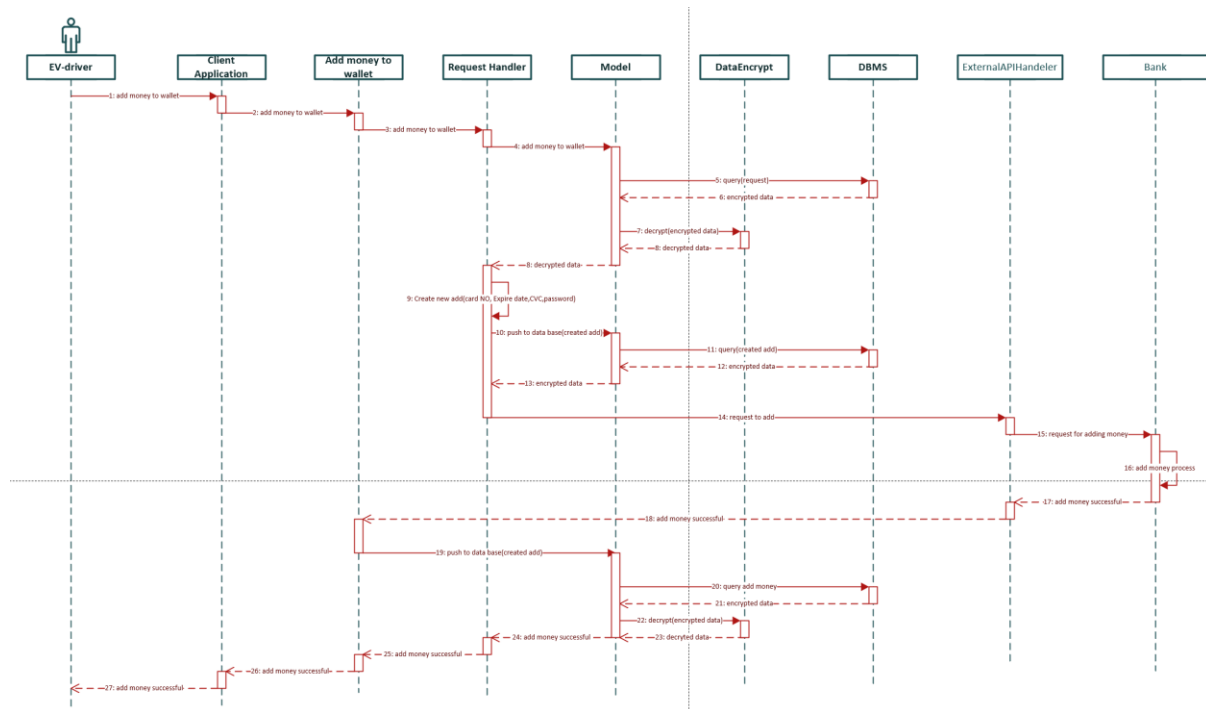
2.4.4. Edit personal information

This diagram is too similar to the Add vehicle specification since both tasks are in relation with EV-driver's information and can be done in the same component named Modify Information. EV-driver new personal information in the client application (EMSP) component. When EV-driver clicks on Save button, the request is then sent to request handler component to check to which component should the request be sent for being handled. Then the request and new user data are sent to Modify Information component. Then this component, sends the information to Model. Model sends a query to database to update the user personal information. Finally, DBMS sends an encrypted result which should be first decrypted through DataEncrypt component. If the retrieved data from DataEncrypt component is not null, it means that the new personal information of the named user has been added to DBMS successfully.



2.4.5. Adding money to wallet

EV-driver clicks to ...

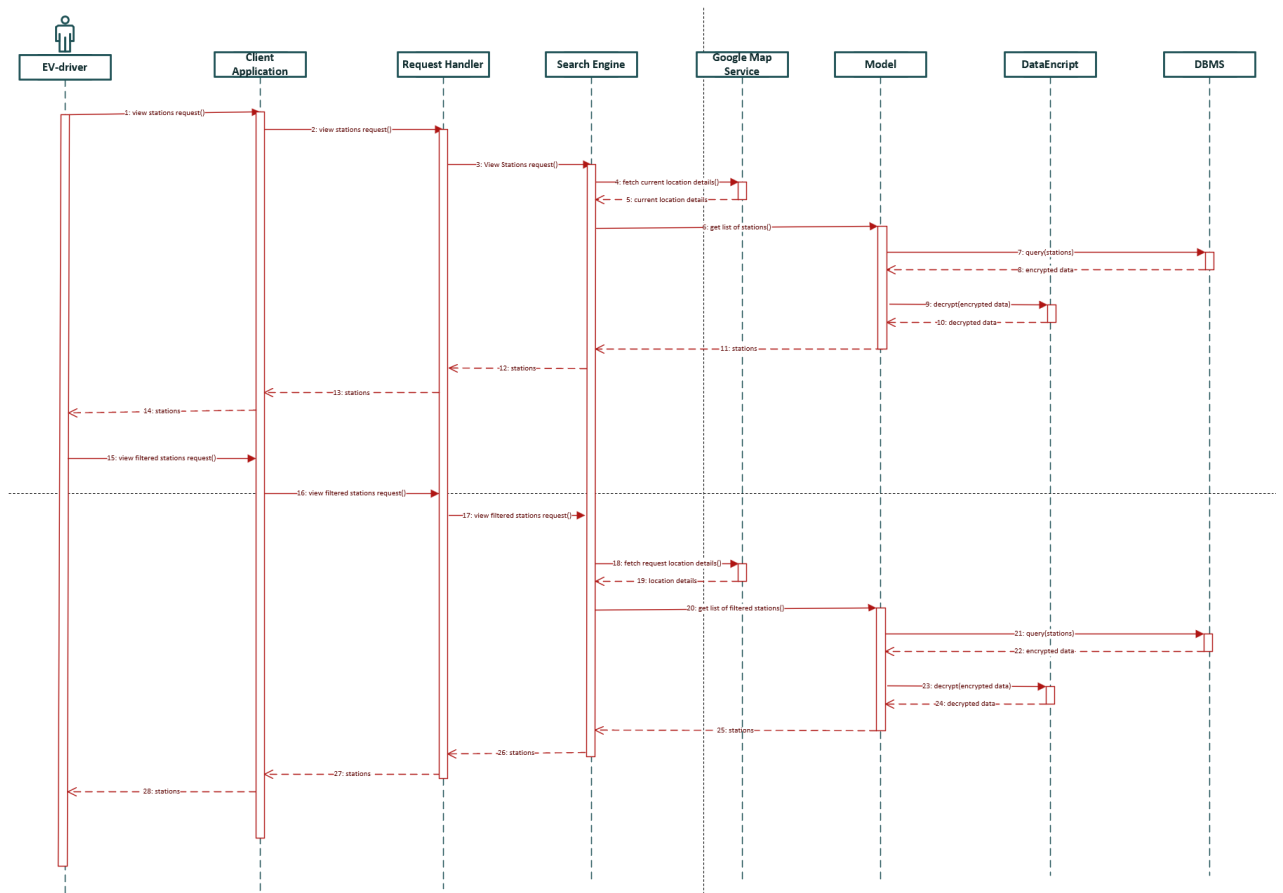


مشکل داره و کامل نیست !!!!!!!!!!!!!

2.4.6. Search

EV-driver clicks view charging stations on client application component. In the first page, the nearby charging stations are shown to the user. So, client application sends the request to “Request Handler” component. Request Handler sends the request to of searching based on distance to the Search Engine component. Now, Search Engine must first get the current location of EV-driver. So, it sends a request to Google Map Service component to get the current location. Then, it is time to check the DBMS for nearby charging stations based on locations details received from Google Map Service component. So, the searching details along with search request is sent to Model component. It sends a query to DBMS. DBMS sends the result encrypted. So, we use the DataEncrypt to decrypt the result of DBMS. Finally, the retrieved decrypted data is the list of nearby charging stations that will be shown to user.

Now, it is time to search for charging stations based on other factors e.g. rate, price, type of socket and etc. Therefore, the above steps must be repeated here again but the searching details can be other factors. So, the only thing that changes here is the arguments of the request sent to Model component for searching in the DBMS.

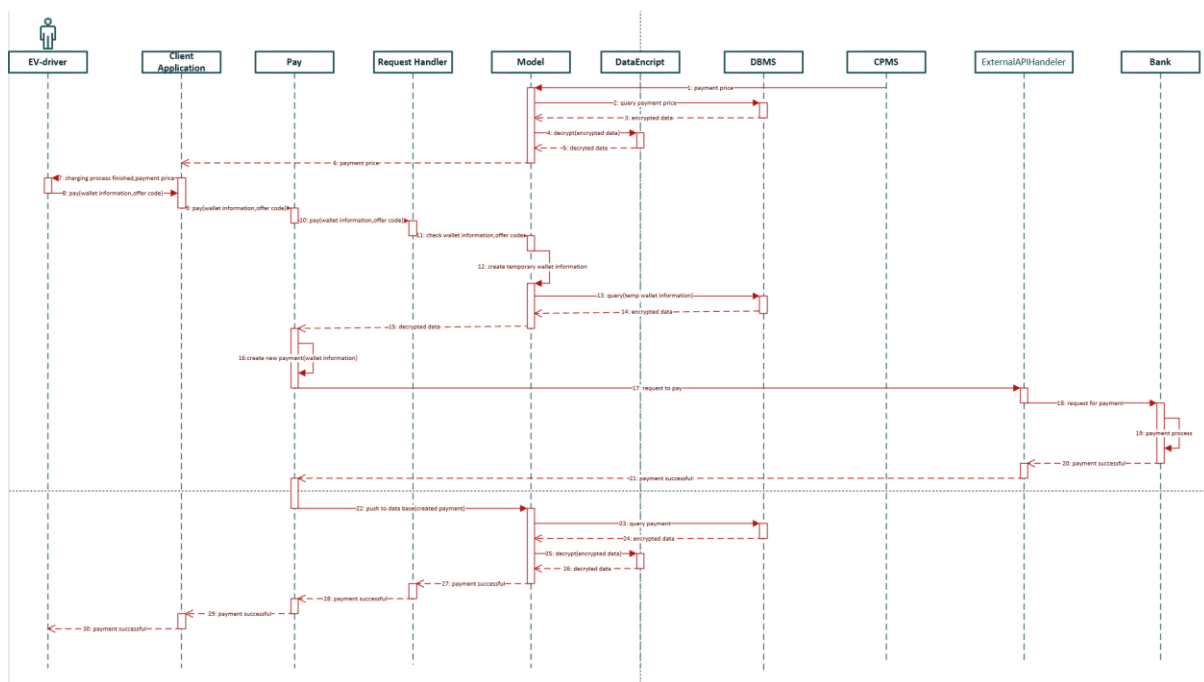


2.4.7. Book

EV-driver clicks to ...

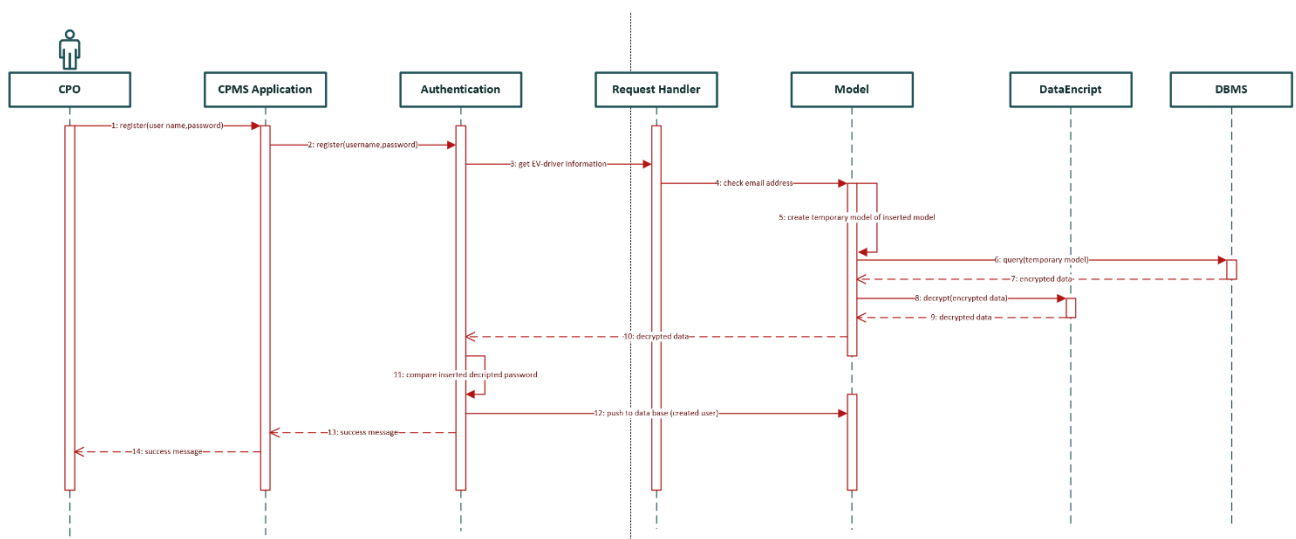
2.4.8. Pay

EV-driver clicks to ...



2.4.10. CPO login

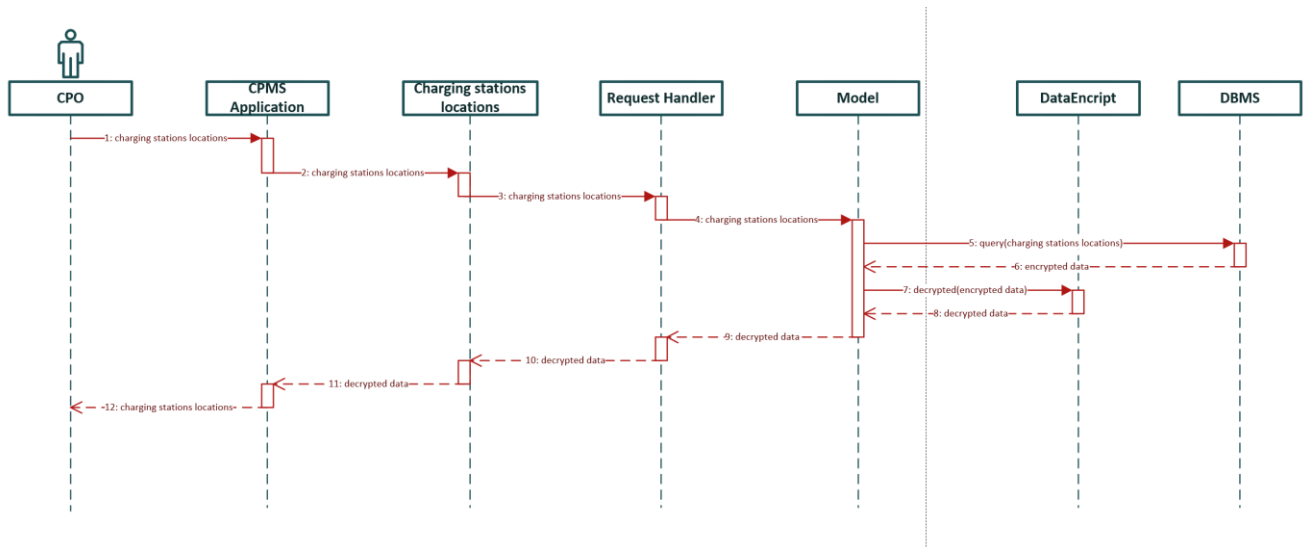
CPO enters email and password through CPMS application component. Then, his/her information is sent to Authentication component. Now, it should be checked if the information entered is correct or not. First the email must be checked to see if it exists in the DBMS or not. Then the password must be checked. This is checked in Model component. So, the Authentication component sends the information to Model component. Model sends a query to DBMS to check if the email exists in the app before or not. The answer of DBMS is encrypted for security purposes. So, there is a DataEncrypt component which decrypts the reply from DBMS. If the retrieved data is null, it means there is no user with the entered email. Otherwise, the user exists. Now, the Authentication component should check if the entered password is the same as decrypted password retrieved from DBMS or not. If the check result is positive, the retrieved user is returned, otherwise a null value is returned.



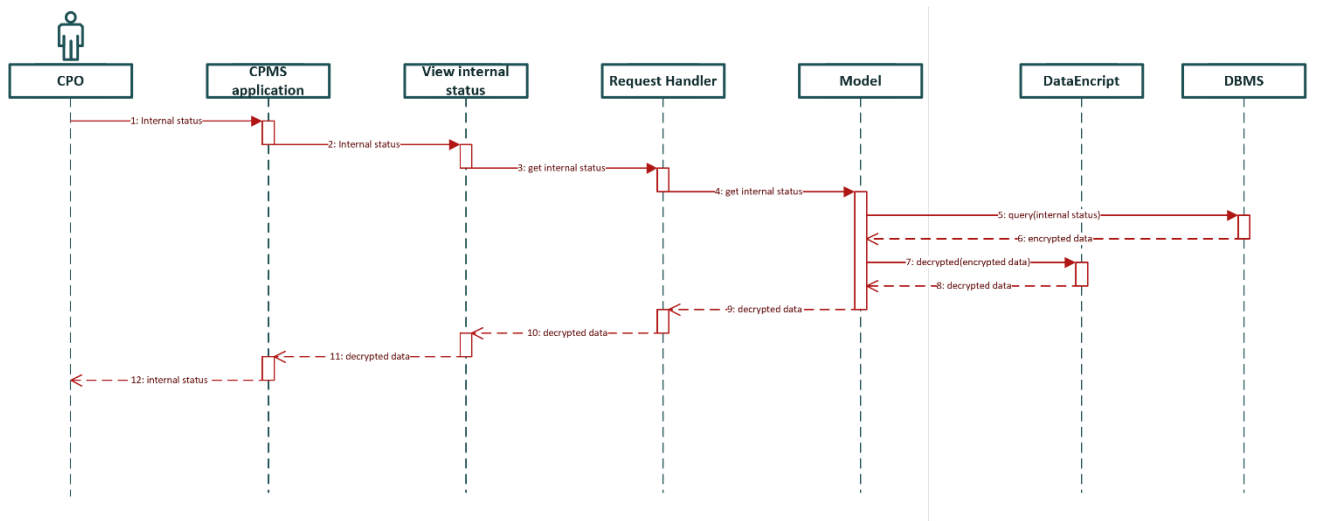
2.4.11. CPO view charging stations' locations

EV-driver clicks to view the locations and details of charging stations through CPMS Application component. The request is then sent to Request Handler component to be sent to the related component for being handled which is View Charging Point Information component. This component is in charge of showing charging point internal and external status and location. All of this information must be got from DBMS. So, a request is sent to Model. Then Model component sends a query to DBMS to get the related information. The data received from DBMS is encrypted and must be decrypted using DataEncrypt component. The data retrieved from DataEncrypt component is the information that must be shown to the CPO through CPMS application component. Therefore, the data is sent back again to all components until CPMS application component.

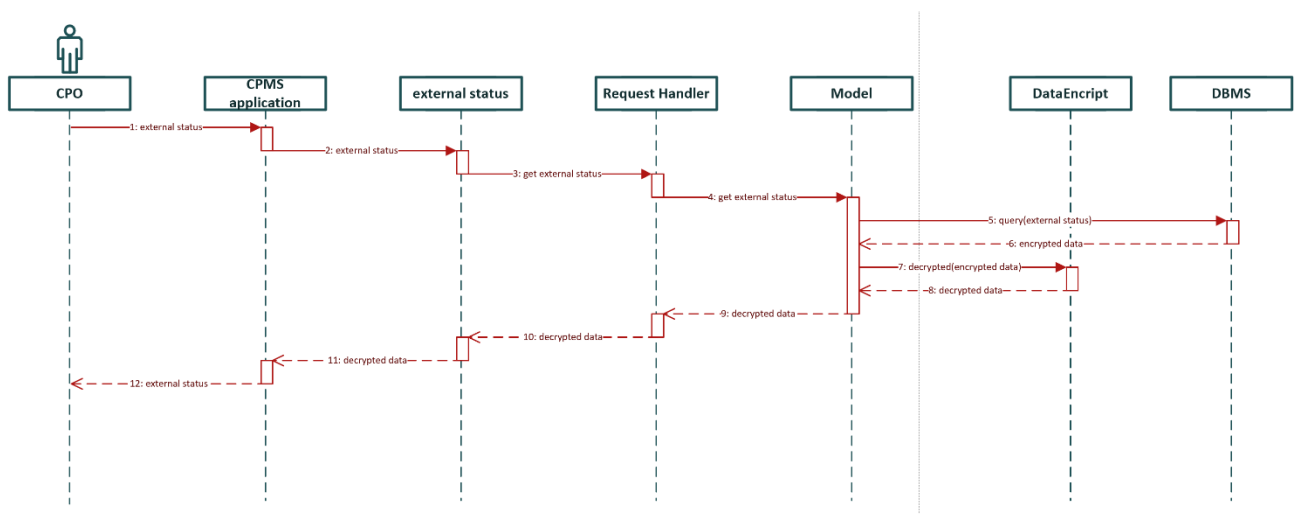
These explanations are the same for the following 2 diagram, viewing internal and external status.



2.4.12. CPO view internal status



2.4.13. CPO view external status



2.5. Component Interfaces

2.6. Selected Architectural Styles and Patterns

2.7. Other Design Decisions

3. User Interface Design

3.1. EV-Driver

3.1.1. Register/Verifying email

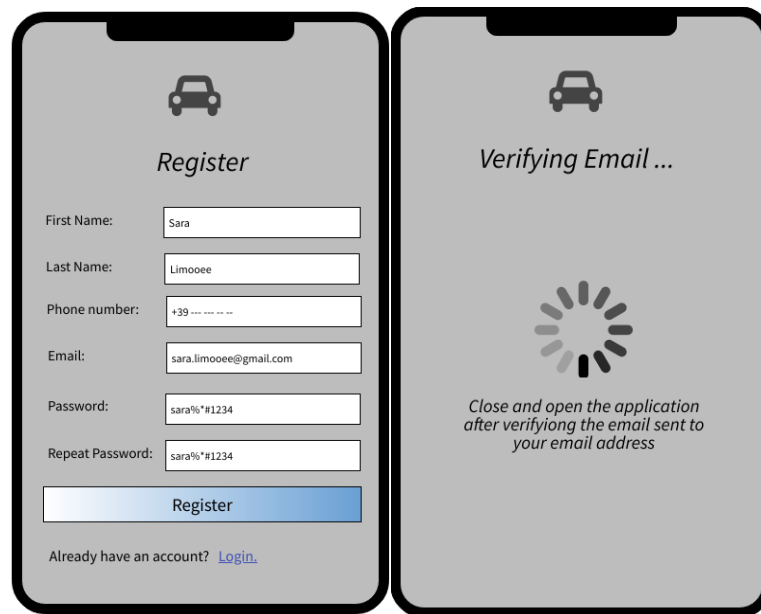


Figure 1: UI for Login and signing in/Verifying email address

3.1.2. Login

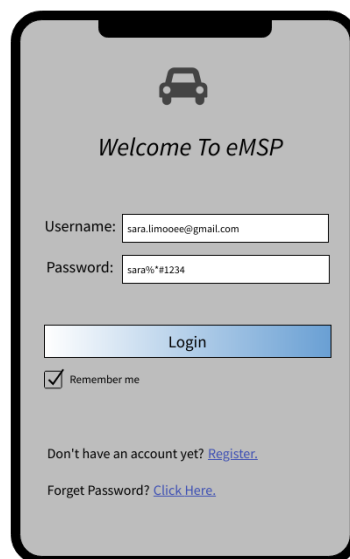


Figure 2: UI for Login

3.1.3. eMSP main page

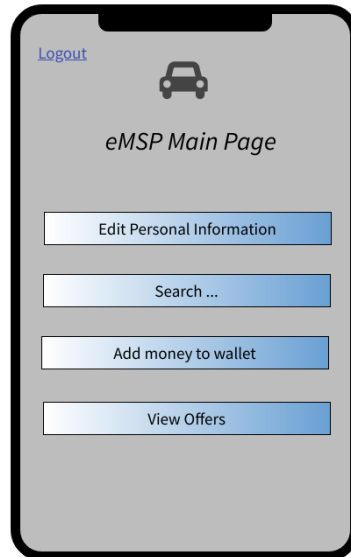


Figure 3: UI for eMSP main page

3.1.4. Add vehicle specification

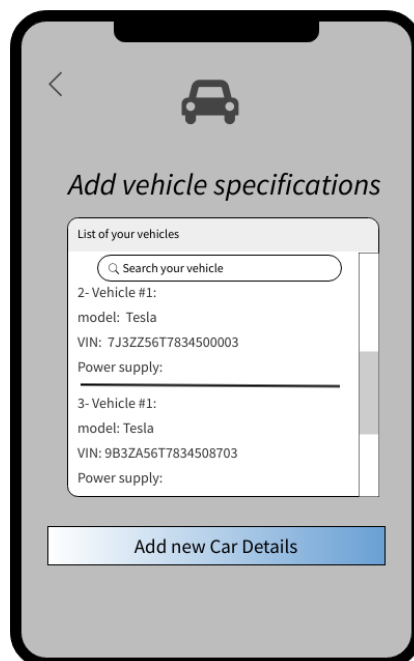



Figure 4: UI for adding vehicle specifications


3.1.5. Edit personal information



The image shows a mobile app interface for editing personal information. At the top, there is a back arrow on the left and a car icon in the center. Below the car icon is the title "Edit Personal Info.". The form contains several input fields: "First Name:", "Last Name:", "Phone number:", "Email:", "New password:", and "Repeat new password:". Each field is represented by a white rectangular box. At the bottom of the form is a blue button with the text "Save".

Figure 5: UI for editing personal information

3.1.6. Add money to the Wallet



The image shows a mobile app interface for adding money to a wallet. At the top, there is a back arrow on the left and a car icon in the center. Below the car icon is the title "Wallet". The form contains several input fields: "Credit:" (a small white box), "Name:" (a white box), "Card no.:" (a white box), "Expire Date:" (a date picker showing "12 May 2016" and a calendar icon), and "CVC:" (a small white box). Below the "Expire Date:" field is a checkbox labeled "Remember my card". At the bottom of the form is a blue button with the text "Book".

Figure 6: UI for adding money to wallet

3.1.7. Search for charging stations/Filter charging stations

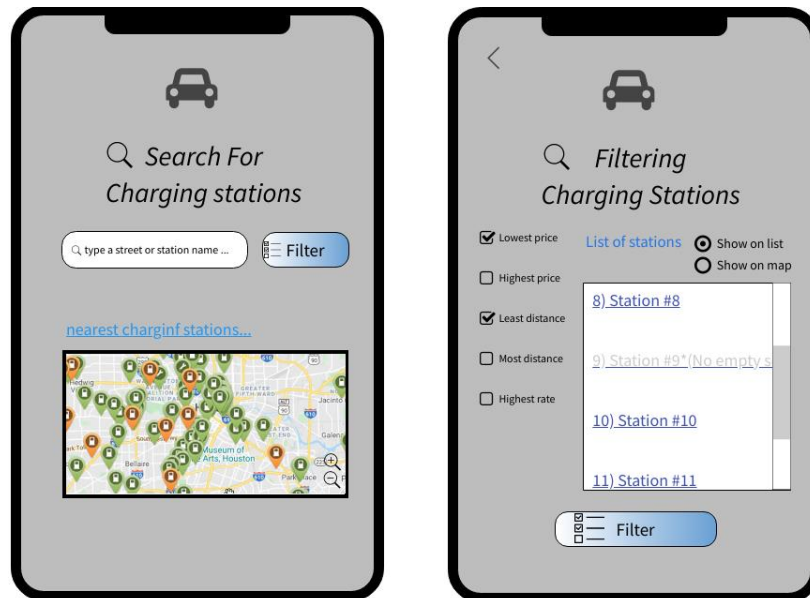


Figure 7: UI for Searching for charging stations/Filtering charging stations

3.1.8. Charging station details

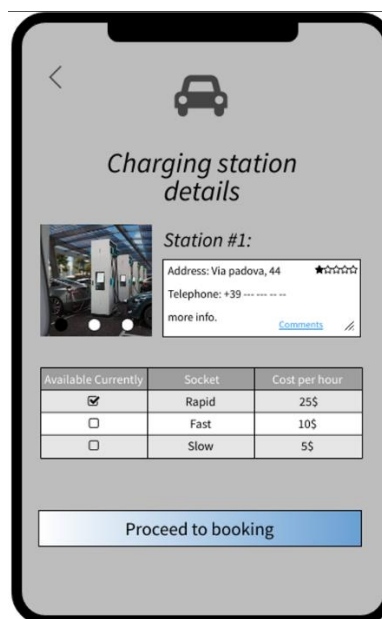


Figure 8: UI for Charging station details

3.1.9. Book/Book receipt

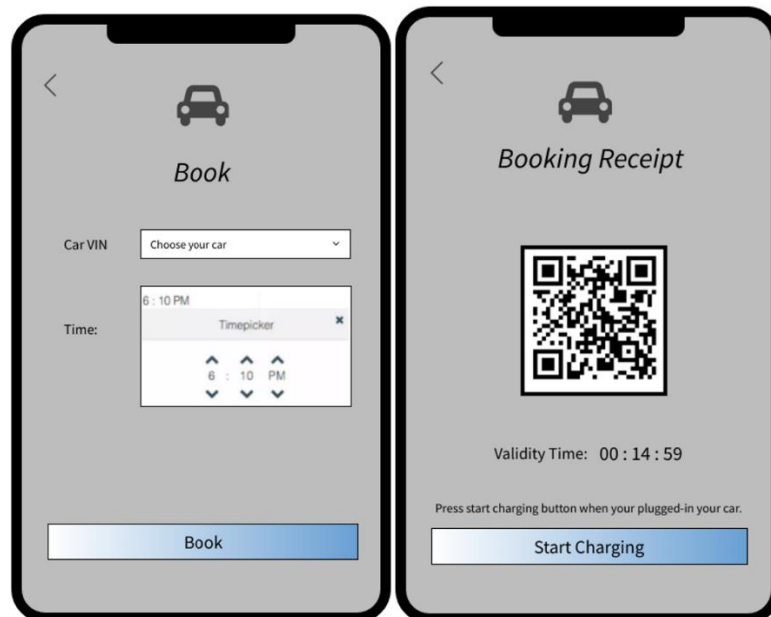


Figure 9: UI for Charging station details/Booking receipt

3.1.10. Charging process/Charging process finished

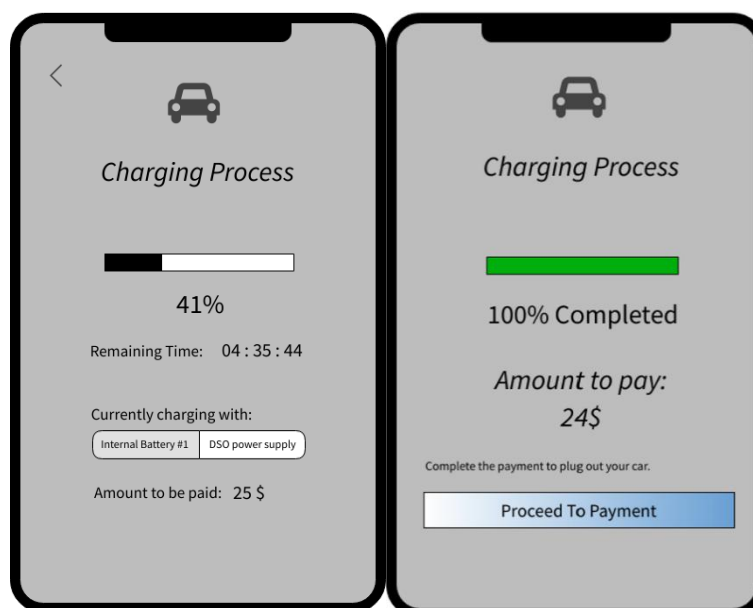


Figure 10: UI for Charging process/Charging process finished

3.1.11. Pay/Rate

The image shows two side-by-side mobile app screens. The left screen is titled 'Pay' and features a car icon at the top. Below the title, it says 'Amount to be paid: 25 \$'. There are two tabs: 'Card details' and 'Wallet'. Under 'Card details', there are input fields for 'Name:', 'Card no.:', 'Expire Date:' (with a calendar icon and '12 May 2016'), 'CVC:' (with '123' entered), and 'Offer code:'. There is a checkbox labeled 'Remember my card' which is checked. At the bottom is a blue 'Pay' button. The right screen is titled 'Rate' and also has a car icon at the top. It features a back arrow in the top left. Below the title, there is a photo of a station and a box labeled 'Station #1:' containing 'Address: Via padova, 44' and 'Telephone: +39 ...'. Below this is a text input field for 'more info.'. A star rating section says '★ Please help us to enhance our services.' with a 'Score*:' label and five stars, the first of which is filled. Below that is a 'Reason:' label and a text input field with the placeholder 'Insert at most 150 characters.'. At the bottom is a blue 'Submit' button.

Figure 11: UI for Paying the bill/Giving a rate to received serviced

3.2. CPO

3.2.1. Login

The image shows a single mobile app screen for the 'CPMS System'. It has a car icon at the top. Below the icon, the title 'CPMS System' is displayed. There are two input fields: 'Username:' with the text 'sara.limooee@gmail.com' and 'Password:' with the text 'sara%*#1234'. Below these is a blue 'Login' button. Under the button is a checkbox labeled 'Remember me' which is checked. At the bottom, it says 'Forget Password? [Click Here.](#)'.

Figure 1: UI for login

3.2.2. Charging stations status/Charging station details

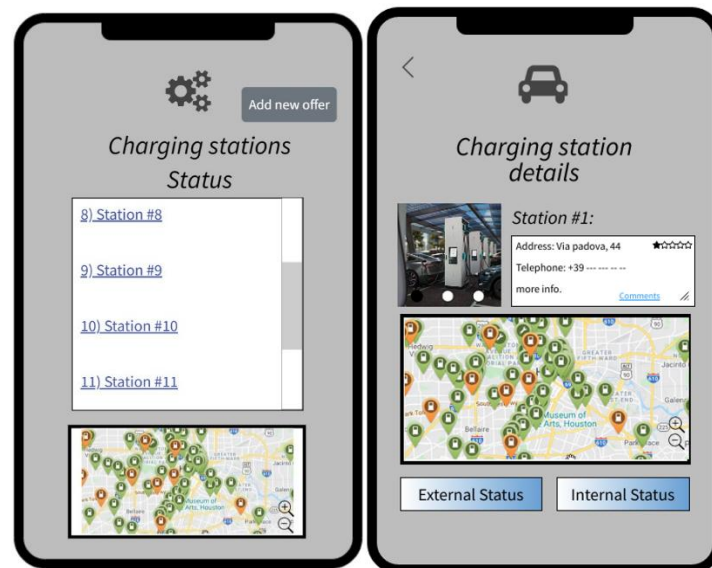


Figure 1: UI Charging stations status/Charging station details

3.2.3. Internal status/charging vehicles

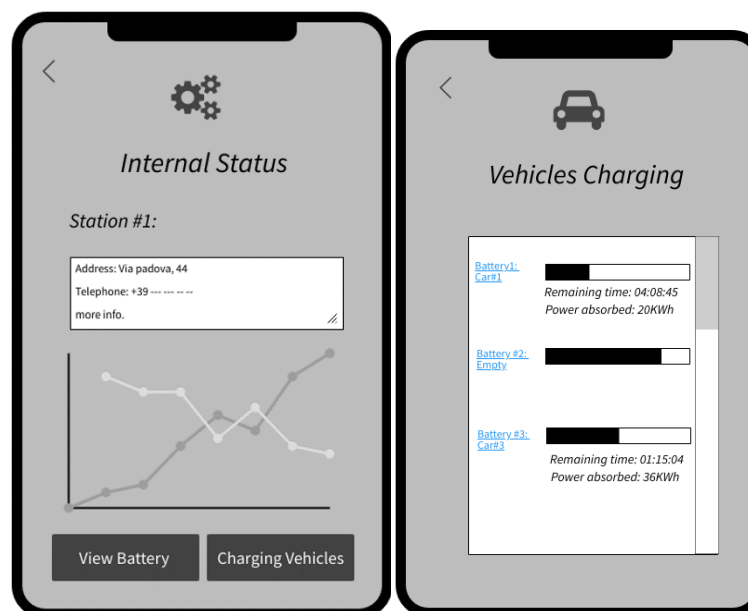


Figure 1: UI for Internal status/charging vehicles

3.2.4. External status/Batteries status

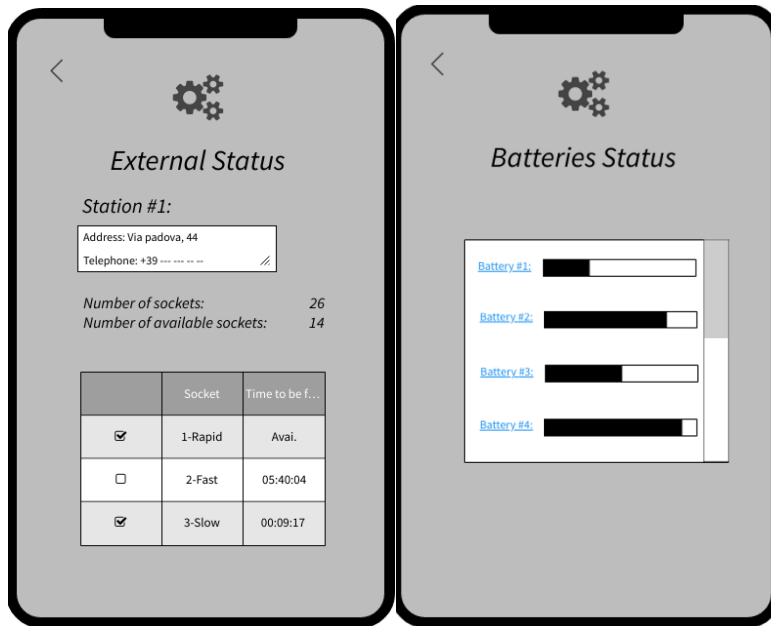


Figure 1: UI for External status/Batteries status

4. Requirement Traceability

In this section, a table is provided in which the components that are required in order to full fill each of the requirements specified in RASD are explained. Furthermore, in order to save space, the abbreviations of the components have been written in the list below.

- EV-driver:
 - EVA: EV-driver application
 - AU: Authentication
 - MA: Mail
 - RH: Router Handler
 - GMS: Google map service
 - SE: Searching
 - BO: Booking
 - PA: Pay
 - EI: Edit information
 - EAPIH: External API handler
 - MO: Model
 - DB: DBMS

- CPO:
 - CA: CPO application
 - AU: Authentication
 - RH: Request Handler
 - VL: View locations
 - VIS: View internal status
 - VES: View external status
 - VDP: View DSO's prices
 - CP: Charging process
 - SDE: Select DSO to acquire energy
 - EAPIH: External API handler
 - MO: Model
 - GMS: Google map service
 - DB: DBMS

- EV-driver

R	UA	AU	MA	RH	GMS	SE	BO	PA	\$EI	EAPIH	MO	DB
R1	*	*		*					*		*	*
R2	*		*	*							*	*
R3	*	*		*							*	
R4		*	*	*						*	*	
R5	*	*	*	*							*	*
R6	*	*		*				*	*	*	*	*
R7	*	*	*	*				*		*	*	*
R8	*	*		*				*	*		*	*
R9	*			*	*	*				*	*	*
R10	*			*	*	*				*	*	*
R11	*	*		*		*	*				*	*
R12	*			*		*				*	*	*
R13	*			*		*				*	*	*
R14	*	*		*		*					*	*
R15	*			*	*	*	*			*	*	*
R16	*			*			*				*	
R17	*			*	*		*			*	*	*
R20	*			*							*	
R23	*	*		*						*	*	*
R25	*	*		*							*	*
R26				*							*	
R28	*	*		*							*	*
R29	*	*	*	*				*		*	*	
R31	*	*	*	*				*		*	*	
R32	*	*	*	*				*		*	*	
R33	*	*	*	*							*	*
R34	*	*	*	*							*	*
R35	*	*		*							*	*
R36	*	*		*							*	*

- CPO

R	CA	AU	RH	VL	VIS	VES	VDP	CP	SDE	EAPIH	MO	GMS??	DB
R18			*	*						*	*		*
R19	*		*	*	*			*			*		
R21	*		*	*	*	*		*		*	*		*
R22	*		*							*	*		
R24			*	*	*			*			*		
R27	*		*			*	*			*	*		*
R30			*							*	*		*
R37	*		*							*	*		*
R38	*	*	*								*		*
R39	*	*	*								*		*
R40	*	*	*	*		*	*		*		*		*
R41	*	*	*	*	*						*		*
R42	*	*	*	*	*						*		*
R43	*	*	*				*			*	*		*

5. Implementation, Integration and Test Plan

5.1. Overview

In this section, the plans for implementation and testing the application are introduced and all preliminary considerations needed to implement and test both EMSP and CPMS applications are presented.

5.2. Implementation Plan

In order to implement both systems with regard to the component diagram introduced in section 2.2., we use a bottom-up approach so that each component can be implemented and tested separately.

The components and subcomponents described in the component view section can be divided as these systems:

- Client application (EMSP)
- CPMS application

Other subcomponents are different parts of these two systems that interact with each other.

Since the implementation of some components depend on some other components, the implementation must be done from the lower components to the top ones. For example, each component and subcomponent, rely on DBMS. So, it should be implemented first. Or, Request Handler component must be implemented before other components since it determines which component should handle each request of users of the system (EV-driver or CPO).

On the other hand, Google Maps and external API Handler are two other components which play a role like database and are main components. So, they should be implemented in this step.

After that we can proceed to the implementation of the subcomponents in each Client Application component and CPMS application component. Here, there are some main subcomponents such as the Model and the DataEncrypt subcomponents. Most components use them as a connection to access to the database.

Then, The Mail Service and Authentication subcomponents turn. The Mail subcomponent should be implemented before the other one, because the functionalities of Authentication rely on Mail Service subcomponent and we want to keep using the bottom-up approach while testing.

5.3. Integration & Test Plan

The sequence in which we test the system will follow the implementation order described in the previous section. The right-to-left order, as seen from the component diagram, corresponds to a bottom-up testing strategy which should work well considering the quite simple hierarchical structure of the system.

```
$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
```

6. Effort Spent

7. References

- Specification Document: “Assignment RDD AY 2022-2023_v2.pdf”
- Course slides