



POILTECNICO

MILANO 1863

e-MALL

e-Mobility for All

RASD

Requirement Analysis and Specification 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 the world warming and the increase in the number of pollutions 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 the 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 develop the application. In general, the main different features of this document are:

- The high-level architecture
- Main components of the system
- Interfaces provide 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 CMPS 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 system when he/she must be notified about something (ex: when a now 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

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.

¹ Layer and tier words can be used interchangeably.

Picture done

2.3. Deployment View

2.4. Runtime View

2.4.1. EV-Driver Register

2.4.2. Login

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

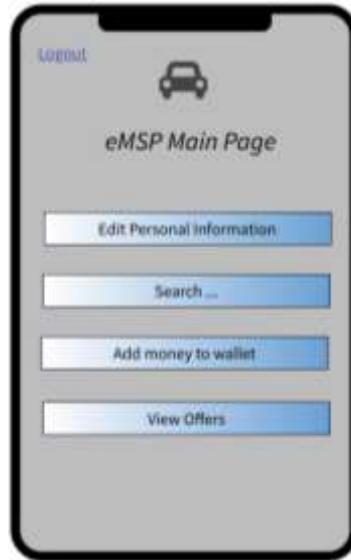


Figure 3: UI for eMSP main page

3.1.4. Add vehicle specification

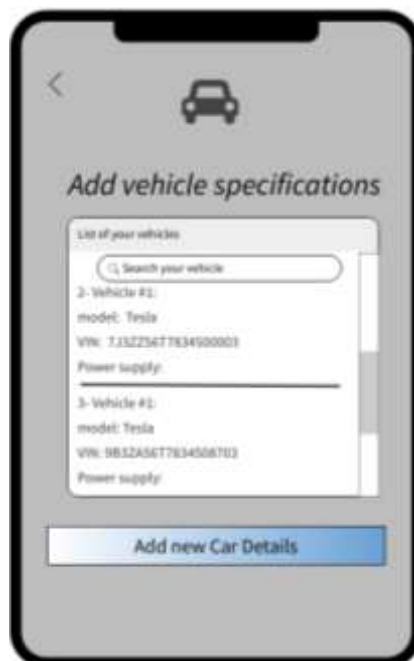
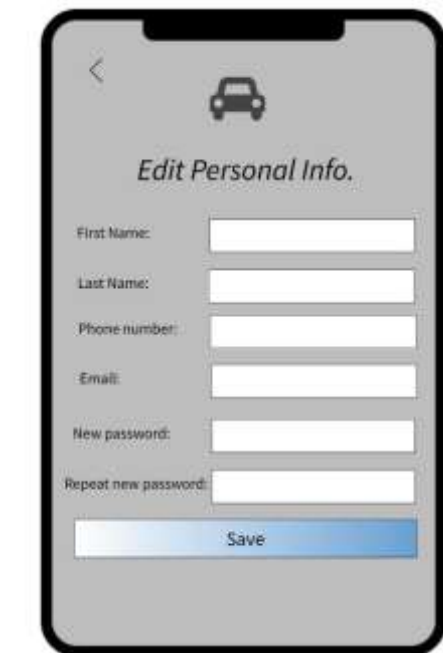


Figure 4: UI for adding vehicle specifications

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 white box), "Name:" (a white box), "Card no.:" (a white box), "Expire Date:" (a date picker showing "12 May 2016"), and "CVC:" (a white box). Below the "Expire Date:" field is a checkbox labeled "Remember my card" which is checked. 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

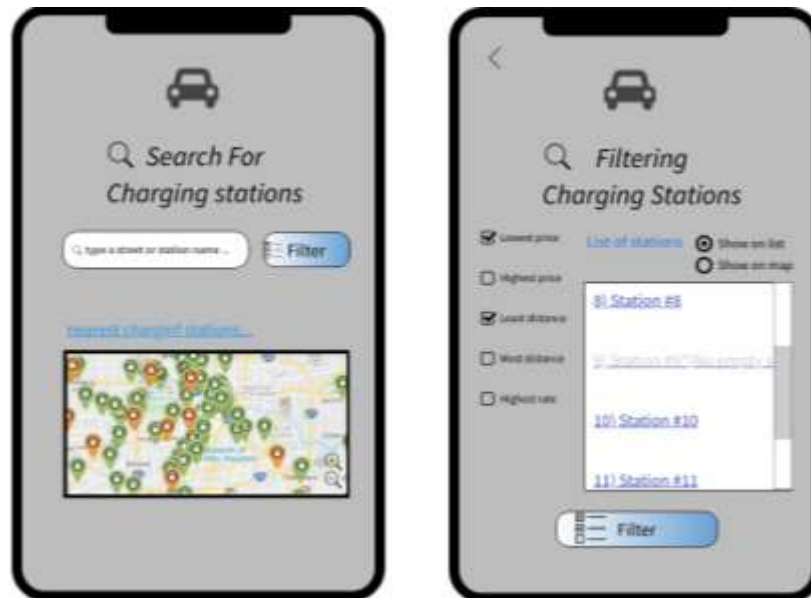


Figure 7: UI for Searching for charging stations/Filtering charging stations

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

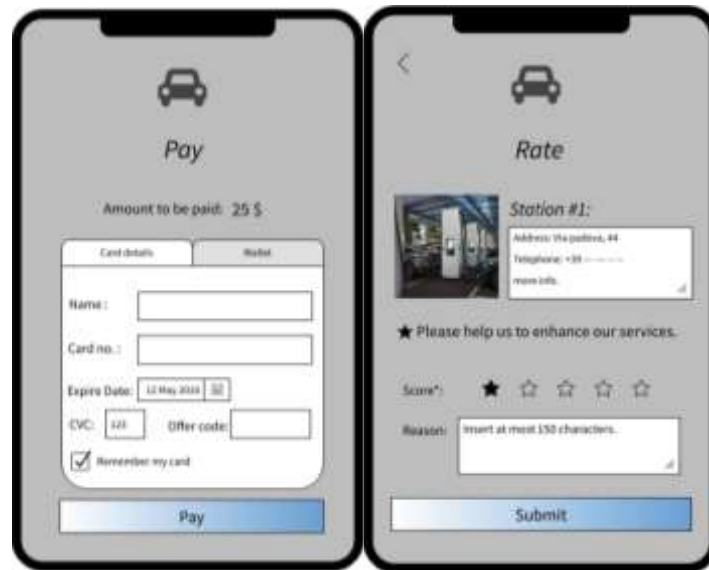


Figure 11: UI for Paying the bill/Giving a rate to received serviced

3.2. CPO

3.2.1. Login



Figure 1: UI for login

3.2.2. Charging stations status/Charging station details

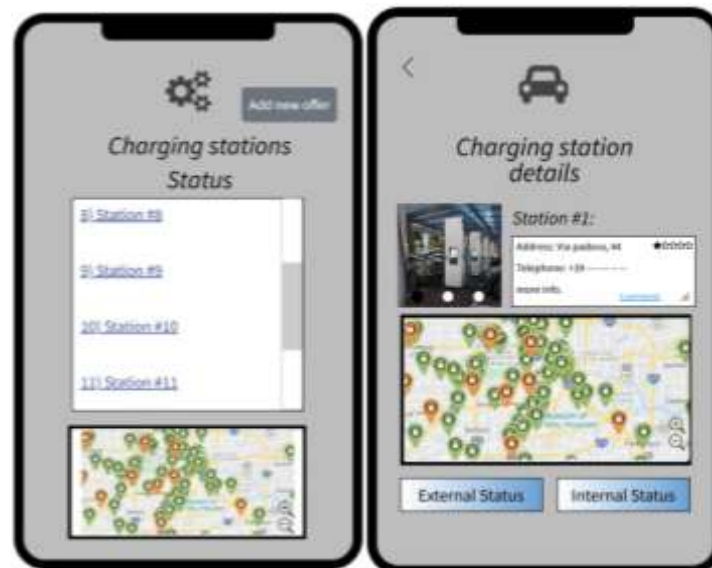


Figure 1: UI Charging stations status/Charging station details

3.2.3. Internal status/charging vehicles

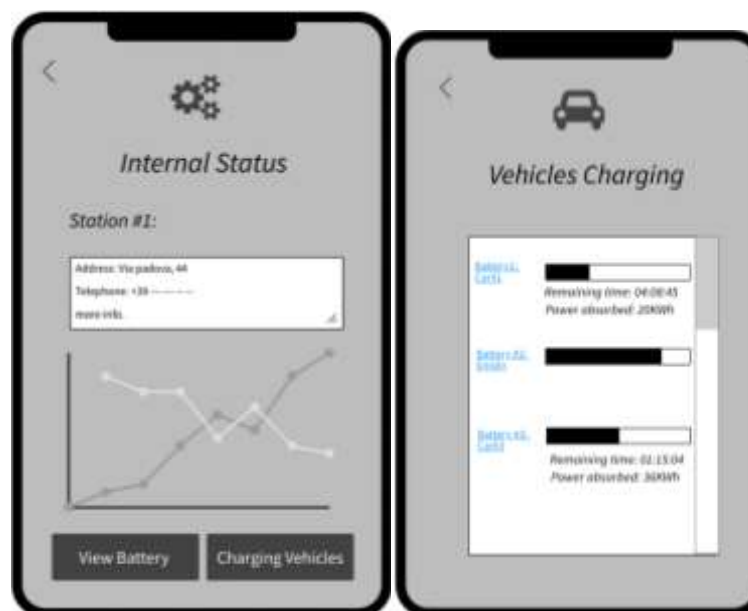


Figure 1: UI for Internal status/charging vehicles

3.2.4. External status/Batteries status

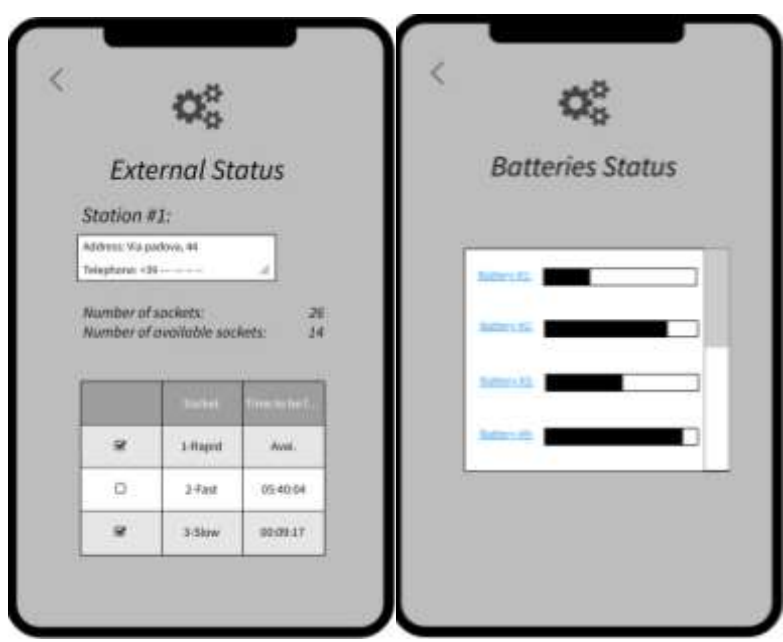


Figure 1: UI for External status/Batteries status

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:
 - UA: User application
 - AU: Authentication
 - MA: Mail
 - RO: Router
 - 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
 - RO: Router
 - 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	RO	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	RO	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. Implementation

5.2. Integration & Test Plan

6. Effort Spent

7. References

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