POLITECNICO DI MILANO

Computer Science and Engineering

Implementation and Test Deliverable

eMall - e-Mobility for All

Software Engineering 2 Project

Academic year 2022 - 2023

5 February 2023

Version 1.0

Authors:
Balestrieri Niccolò
Bertogalli Andrea
Tombini Nicolò

Professor:
Matteo Camilli

Version history

Date	Revision	Notes
05/02/2023	v.1.0	First release.

Contents

Chapter 1

Introduction

1.1 Purpose

The goal of **I**mplementation **a**nd **T**est **D**eliverable Document lies in a meticulous explanation of the whole developping process about eMall system according to what was defined in the previous document, **RASD** and **DD**. The audience of the presented document is addressed to investors that deal with investments in rinnovable energies' field.

1.2 Scope

Obviously, the main scope of this document is to provide a fully detailed explanation of what was developed, from all the implemented and released software features to unit testing of each module.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Term	Definition
End user	Identifies electric vehicles owners who use the service, can also be referred as user.
Charging station	Place of charging of electric vehicles.
Charging Station available	At the moment at least one socket is free.
Charging Station not available	At the moment all the sockets are occupied.
Charging socket	Single charging point.
Booking	Reserved time slot generated by the system for charging vehicles. It also can be referred to as Reservation.
Time slot	Slot of time in which customers can charge their vehicles.
QR-Code	A QR code is a type of matrix barcode (or two-dimensional barcode).
System	Set of hardware and software tools, that provide the desired service. It can be considered as eMall, Application and Platform.
Internal status	For internal status of a charging station: the amount of energy available in its batteries, the number of vehicles being charged and, for each charging vehicle, the amount of power absorbed and time left to the end of the charge.
External status	For external status of a charging station: the 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.
CPO	The charging station's administrator.
WebSocket	WebSocket is a full-duplex communication protocol that enables bi-directional communication between a client and a server over a single, long-lived connection

1.3.2 Acronyms

Acronyms	Meaning
eMall	e-Mobility for All
eMSP	e-Mobility Service Provider
CPO	Charging Point Operator
CPMS	Charging Point Management System
DSO	Distribution System Operator
API	Application Programming Interface
GPS	Global Positioning System
HTTPS	Hyper Text Transfer Protocol Secure
HTTP	Hyper Text Transfer Protocol
SHA	Secure Hash Function
DBMS	Database management system

1.3.3 Abbreviations

Abbreviations	Meaning
e.g.	exempli gratia
etc.	Etcetera

1.4 Reference Documents

- *Course slides on WeeBeep.*
- *DD eMall system.*
- *RASD eMall system.*
- *I&T assignment document.*

1.5 Document Structure

This section presents the structure of the document:

1. **Introduction**: this is the first section and provides an overview of the entire document.
2. **Implemented Requirements**: this section should provide a description of what was implemented and what not.
3. **Adopted Development Framework**: this section details the framework used for development.
4. **Source Code Structure**: this section outlines the structure of the source code both in the front and end in the back end.
5. **Testing**: this section describes the testing process and results.
6. **Installation Instructions**: this section provides instructions on how to install the software.
7. **Effort Spent**: this section outlines the amount of time and resources spent on the project.
8. **References**: this section lists any references used in the document.

Chapter 2

Implemented Requirements

The following requirements have been implemented in the prototype; they were chosen as they play a crucial role in appealing to potential users of the system.

Requirement	Description	Implemented
R.1	The system must allow the end user to register himself to the system by providing all the mandatory information.	Yes
R.2	The system must allow the end user to register a payment method.	Yes
R.3	The system must verify the uniqueness of the mail and username of the end user to be registered.	Yes
R.4	The system must send a confirmation on the end user's email after his registration.	No
R.5	The system must allow the end user to log into his account by entering his username and password.	Yes
R.6	The system must allow the end user to book a charging socket of a certain charging station in a specific time slot.	Yes
R.7	The system must generate a QR-Code after the end user's reservation.	Yes
R.8	The system must show to the end user the suggested charging stations.	Yes
R.9	The system must show to the end user the available and unavailable time slots of a certain charging socket.	Yes
R.10	The system must access to the end user's location through GPS.	Yes
R.11	The system must access to the end user's calendar.	No
R.12	The system must access to the end user's vehicle status battery.	No
R.13	The system must allow the end user to delete his reservation.	Yes

R.14	The system must notify the end user when the charge of his vehicle is completed.	Yes
R.15	The system must subtract the cost of the charge from the end user's payment method.	No
R.16	The system must send to the end user an email with the invoice for the used service.	No
R.17	The system must allow the end user to book a time slot.	Yes
R.18	The system must stop the charge when the upper bound of timeslot is exceeded.	No
R.19	The system must allow the end user to see his current bookings.	Yes
R.20	The system must allow the end user to search for a certain charging station.	Yes
R.21	The system must allow the end user to stop the charging process.	Yes
R.22	The system must allow the end user to start the charging process.	Yes
R.23	The system must allow the end user to monitor the charging process.	Yes
R.24	The system must prevent the end user from reserving a charging socket in a certain time slot, if he already has another reservation in that time slot.	Yes
R.25	The system must prevent at least two different end users from reserving the same charging socket in the same time slot.	Yes
R.26	The system must allow the CPO to register himself to the system by providing all the mandatory and commercial information.	Yes
R.27	The system must verify the uniqueness of the mail of the CPO to be registered.	Yes
R.28	The system must verify the correctness of the CPO's commercial data.	Yes
R.29	The system must send a confirmation on the CPO's email after his registration.	No

R.30	The system must allow the CPO to log into his account by entering his username and password.	Yes
R.31	The system must allow the CPO to know the internal and external status of charging sockets of his charging stations.	Yes
R.32	The system must allow the CPO to monitor the charging process.	Yes
R.33	The system must allow the CPO to acquire information about the price of energy from DSOs.	Yes
R.34	The system must allow the CPO to acquire energy from DSOs.	Yes
R.35	The system must allow the CPO to decide whether or not to store or not energy.	No
R.36	The system must allow the CPO to decide whether to use available energy in the batteries.	Yes
R.37	The system must forbid to charge the vehicle to a user that scans a QR-Code which is not valid (expired, wrong etc.).	Yes
R.38	The system must indicate whether a slot is booked or not, and it must forbid to book an already booked time slot.	Yes
R.39	The system must manage the concurrent booking of a slot: if a user waits too much to book a slot and meanwhile another user books that slot, the system must notify the first user when he tries to perform the booking.	Yes

R.40	The system must allow the CPO to add a new charging station.	Yes
R.41	The system must allow the CPO to view, via the CPMS, the charge percentage of the battery cluster.	Yes
R.42	If the user initiates the charging and there is not enough charge in the battery cluster, the system, to avoid service disruption, allows the charging station to receive power directly from the DSO channel.	Yes

Chapter 3

Adopted Development Framework

Below, are described firstly the frameworks used and then the most import modules of the three software implemented.

3.1 Frameworks

- **React**: React.js is a JavaScript library for building user interfaces. It was developed by Facebook and is now maintained by Meta and a community of individual developers and companies. React allows developers to build large, complex user interfaces by breaking them down into smaller, reusable components. Each component can manage its own state, making it easier to build and maintain large applications. React uses a virtual DOM (Document Object Model) to optimize updates and rendering, making it fast and efficient. React can be used with a variety of programming languages and tools, making it a versatile choice for building web applications. It is widely used for building single-page applications, mobile apps, and progressive web apps.
- **NodeJS**: NodeJS is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It is commonly used for building server-side applications.
- **Flutter**: Flutter is an open-source mobile application development framework created by Google. It uses the Dart programming language and offers a fast, expressive, and flexible way to build natively compiled applications for mobile, web, and desktop from a single codebase. With its rich set of customizable widgets and tools, Flutter makes it easy for developers to create high-performance and visually appealing apps that run smoothly on both Android and iOS platforms.

3.2 Modules

- **Chai**: Chai is a JavaScript BDD/TDD assertion library for node and the browser that can be paired with any testing framework. It provides a clean and simple interface for making assertions and can be used for both unit and integration tests. Chai provides a large number of assertions for different types of values and objects, making it a versatile and comprehensive tool for testing.
- **Mocha**: Mocha is a JavaScript test framework running on Node.js. It provides a simple and flexible API for writing and executing tests for Node.js applications. Mocha tests

can be run in series or parallel, and it supports asynchronous testing, which allows for testing code with async functions. Mocha also offers features such as reporting, test retries, and test hooks.

- **Socket.io**: Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bidirectional communication between web clients and servers. With Socket.IO you can emit and receive events in real-time, allowing for the creation of real-time applications such as chat rooms, online games, and live-updating pages. The library abstracts the low-level details of WebSockets and provides a simple, event-driven API for working with real-time data. Both the Socket.io server and client were used for development.

- **MUI**: Material-UI is a popular React UI library that implements Google's Material Design specifications. It provides a set of components that follow Material Design guidelines and can be used to build modern and responsive user interfaces. The library provides a wide range of pre-built components for various UI elements such as buttons, cards, icons, forms, and more. The components are highly customizable, allowing developers to style them to match the look and feel of their application. Material-UI also provides support for theming, making it easy to change the color palette and other visual styles across an entire application. Overall, Material-UI is a useful library for developers looking to build beautiful and functional user interfaces with React.

- **Axios**: Axios is a popular JavaScript library that enables communication with HTTP servers, making it easy to retrieve and send data over the internet. It provides a simple, promise-based API for making HTTP requests, including GET, POST, PUT, DELETE, and others. Axios supports all modern browsers and is designed to work with both Node.js and web browsers. Axios provides several features that make it a popular choice for making HTTP requests, including automatic transforming of JSON data, support for canceling requests, and an ability to request data from multiple sources. Axios is often used for building web and mobile applications, particularly with JavaScript frameworks such as React.

- **CryptoJS**: CryptoJS is a library of cryptographic algorithms written in JavaScript. It provides a wide range of cryptographic functions, including AES encryption, SHA hashes, HMAC, Base64 encoding and decoding, and more. The library can be used in web browsers as well as with Node.js. CryptoJS is designed to provide simple, high-level APIs for developers, abstracting away the underlying cryptographic details. It is a widely-used library for adding security to web applications, and is especially useful for implementing encryption and decryption in the browser, where direct access to the operating system's cryptographic libraries is not available. Overall, CryptoJS is a valuable tool for developers looking to add cryptography to their web applications.

- **OpenGeocoder**: OpenGeocoder is a Node.js library that provides geocoding and reverse geocoding functionalities. Geocoding is the process of converting an address or place name into coordinates, while reverse geocoding is the process of converting coordinates into a human-readable address. OpenGeocoder can be used to convert addresses into latitude and longitude coordinates and vice versa, making it useful for mapping applications and location-based services.

- **Google_maps_flutter**: Google_maps_flutter is a Flutter library for integrating Google Maps in mobile applications. It provides a high-level API for displaying maps and markers, allowing developers to easily add Google Maps functionality to their Flutter applications. The library provides a number of customization options, including the ability to change the map's style, add markers, and handle map gestures. With Google_maps_flutter, developers can create engaging and interactive maps experiences with just a few lines of code.
- **http**: The "http" library in Flutter is a Dart package for making HTTP requests. It allows developers to send and receive data from web servers over the internet. The library provides a simple and convenient API for sending HTTP requests, including support for both GET and POST methods. Additionally, the library can handle common HTTP responses, such as redirects and error codes, making it easy to add network communication to a Flutter app. With the "http" library, developers can quickly and easily access RESTful APIs and other web services, making it a valuable tool for building robust and connected mobile applications.
- **Pretty_qr_code**: pretty_qr_code is a Flutter library for generating and displaying visually appealing QR codes. It offers a number of customization options for changing the appearance of the QR code, including the ability to set the color, add a logo, and adjust the border size. This library provides an easy-to-use and flexible API for generating QR codes, allowing developers to quickly and easily add QR code functionality to their Flutter applications. With pretty_qr_code, developers can create eye-catching QR codes that stand out and are easy to scan, making it a useful tool for implementing QR code-based features in mobile apps.
- **ZXing**: ZXing is an open-source, multi-format 1D/2D barcode image processing library implemented in Java. It is widely used for reading and generating QR codes, as well as other barcode formats such as UPC-A, EAN-8, and more. The library has been ported to several other programming languages, including JavaScript, allowing it to be used in web applications. ZXing provides a simple and efficient API for reading barcodes from images, making it a popular choice for developers building barcode scanning applications. The library can be integrated with a variety of platforms, including Android, iOS, and the web, and is well-documented, making it easy to use and customize.

3.3 Programming languages

3.3.1 JavaScript

JavaScript is a high-level, dynamic, and interpreted programming language. It is widely used for creating interactive web applications and front-end development. Some of its advantages include:

- **Easy to learn**: JavaScript has a simple syntax and is easy to learn for developers who have prior programming experience.
- **Versatility**: JavaScript can be used for a wide range of tasks, including server-side programming, desktop application development, and mobile app development.
- **Popularity**: JavaScript is one of the most popular programming languages and has a large community of developers who create and share useful resources.
- **Interactivity**: JavaScript allows for the creation of interactive elements on websites, such as animations, pop-up windows, and more.

However, JavaScript also has some disadvantages, including:

- **Lack of security**: JavaScript is often used for client-side scripting, which makes it vulnerable to security threats such as cross-site scripting attacks.
- **Browser compatibility issues**: Different browsers have different levels of support for JavaScript, which can lead to compatibility issues.
- **Performance limitations**: JavaScript can become slow when used for intensive tasks or when working with large amounts of data.

In conclusion, JavaScript is a popular and versatile programming language that is widely used for front-end development. While it has its disadvantages, its benefits make it an attractive choice for many developers and projects.

3.3.2 Dart

Dart is an open-source, statically typed programming language developed by Google. It is used for building web, server, and mobile applications. Some of the benefits of using Dart include:

- **Easy to learn**: Dart has a syntax that is easy to understand for developers who have experience with other programming languages.
- **Scalability**: Dart is designed for large-scale applications and has features that support scalability and maintainability.
- **Fast performance**: Dart compiles to native code, which results in fast performance and reduced latency compared to other interpreted languages.

- **Cross-platform development**: Dart can be used to build applications for the web, mobile devices, and servers, making it a good choice for cross-platform development.

However, Dart also has some drawbacks, including:

- **Limited adoption**: Dart is not as widely used as other programming languages and has a smaller community of developers.
- **Steep learning curve**: While Dart's syntax is easy to understand, its features and structure can be difficult for new developers to grasp.
- **Lack of support for some libraries**: Dart's relatively limited popularity has resulted in limited support for certain libraries and frameworks.

In conclusion, Dart is a powerful language that is well-suited for building large-scale applications and for cross-platform development. While it has its limitations, its benefits make it a good choice for some projects and developers.

3.3.3 Java (for mobile development)

Java is a popular programming language that is widely used for mobile app development. It is an object-oriented language known for its robustness, security, and portability, making it well-suited for developing complex mobile applications that run on a variety of platforms. Java is the primary language for developing Android apps, and it can also be used to build cross-platform apps that run on multiple mobile operating systems. With its rich ecosystem of libraries and tools, Java provides developers with a wide range of options for building mobile apps. Whether you're building an Android app or creating a cross-platform mobile app, Java is a versatile and powerful language that can help you bring your ideas to life. Advantages of using Java for mobile development:

- **Cross-platform compatibility**: Java is platform-independent and can run on multiple platforms, making it a great choice for cross-platform mobile development.
- **Large development community**: Java has a large and active development community, which means that developers can access a wealth of resources, such as libraries, tools, and support forums.
- **Robust libraries**: Java has a wide range of libraries and frameworks for mobile development, making it easier to add functionality and speed up development time.
- **Security**: Java is known for its security features, making it a good choice for developing applications that handle sensitive data.
- **Performance**: Java is known for its performance, making it suitable for demanding mobile applications, such as games and augmented reality applications.

Disadvantages of using Java for mobile development:

- **Resource-intensive**: Java applications can be resource-intensive, which can lead to slow performance on older and lower-end mobile devices.

- **High memory usage**: Java applications can use a lot of memory, which can be a problem on mobile devices with limited memory.
- **Fragmentation**: Java can suffer from fragmentation, with different versions of the platform running on different devices, which can lead to compatibility issues.
- **Debugging and maintenance**: Debugging and maintaining Java code can be more difficult than with other mobile development platforms, due to its complexity and verbosity.

3.3.4 SQL

SQL (Structured Query Language) is a standard programming language used for managing and manipulating data stored in relational databases. It is used to insert, update, and retrieve data from a database, as well as to create, modify, and delete database tables and relationships between them. SQL is widely used in enterprise applications, web applications, and data analysis, making it an essential tool for many organizations and developers. With its powerful query and data manipulation capabilities, SQL enables efficient and effective management of large amounts of structured data.
Advantages of using SQL:

- **Widely used and supported**: SQL is a widely used and well-established standard, and is supported by a wide range of relational database management systems (RDBMS), including MySQL, Microsoft SQL Server, Oracle, and others.
- **Efficient data management**: SQL provides a powerful and flexible means of managing large amounts of structured data, with efficient querying, insertion, updating, and deletion capabilities.
- **High-level abstraction**: SQL provides a high-level abstraction of the underlying data, making it easier to work with data without having to understand the underlying physical storage structures.
- **Strong data integrity and security**: SQL provides strong data integrity and security features, such as constraints, transactions, and access controls, making it a good choice for applications that handle sensitive data.
- **Good scalability**: SQL databases can be scaled to handle large amounts of data and users, making them a good choice for applications with a high volume of data or users.

Disadvantages of using SQL:

- **Limited scalability for certain use cases**: While SQL is generally good at scaling, it can be limited for certain use cases, such as big data and real-time analytics.
- **Rigid data structure**: SQL requires a rigid data structure, which can be inflexible and make it difficult to handle changing requirements.
- **Performance overhead**: SQL can have performance overhead, particularly for complex queries or large amounts of data.
- **Limited expressiveness**: SQL is limited in terms of expressiveness, compared to general-purpose programming languages, making it less suitable for certain use cases.

Chapter 4

Source Code Structure

4.1 Server Side

The server side is divided in two parts, one handles the HTTP RESTful APIs while the second one manages the WebSockets event. The HTTP part is composed by three js modules: the **EmspHanlder** which handles the eMSP queries, the **CmpsHandler** which handles the CPMS queries and the **SharedHandler** which handles the common APIs between the emsp and cpms such as the login and the registration.

4.1.1 EmspHandler

This module handles all the APIs related to the eMSP, we can categorize them essentially into three categories:

- **Maps APIs**: The emspHandler includes the interaction with external maps api which are helpful to convert some geo-related data that help the eMSP to show the various charging stations on the map.
- **Bookings APIs**: The main part of the emspHandler is composed by booking managment APIs, in fact these APIs allow the eMSP to perform all the operations on the user's bookings.
- **Personal APIs**: As for the CPO also the End user can modify his personal data, so various APIs with this purpose are provided.

4.1.2 CpmsHandler

This module contains the APIs that are used exclusively by the CPMS system. In general we can find three categories of APIs:

- **Charging station APIs**: These are the APIs used by the CPMS to manage the stations.
- **DSO APIs**: These are the APIs used by the CPMS to interact with the DSO (the DSO is simulated).
- **CPO profile APIs**: These are the APIs used by the CPMS to manage the CPO account data.

4.1.3 WebSocketHandler

This module is related with the eMSP, the CPMS and also with the charging socket application. It manages all real time connection, and with the usage of "Rooms" it avoid broadcasting and can redirect all the massages to the desired receiver. Also this module simulates the charging process which, due to obvious infrastructural limitations, has not been fisically implemented. This module in general handles two processes:

- **Booking verification process**: In this case the WebSocketHandler handles the event triggered by the charging socket application when a user scans the QRcode (phisically), then the user application listens for the confirmation event. The process is designed in such a way that only the desired user receives the confirmation message, this is possible thanks to the concept of "Rooms" implemented in the Socket.io library.
- **Vehicle charging process**: This process is more complex, in fact the server must simulate the charging process by keeping synced the information between the CPMS (if someone is monitoring the station) and the end user. To do this several mechanisms have been implemented. Also the server must grant the charging service when the CPO is away.

4.2 Client side

4.2.1 CPO App

The CPO app is developed entirely using React.js and is composed almost only by components in the **src**/**Components** folder. The main components are the five main menus (BatteryMenu, DSOMenu, HomeMenu, StationsMenu, UserMenu) and also the two components that represent the login and the registration. All the other components are only auxiliaries. In the **src** folder we have also the App component which is the entry point for the execution of the applications. Another key element of the application are the Contexts (**src**/**Contexts**) which are very usefull to manage the data access and scope in the whole application.

4.2.2 Charging socket interface

This App is a simple android application that simulates the charging socket, it only implements a WebSocket client which emits event to the server to notify that a user has just scanned a QRCode. The application contains only two files that are respectively the two menues available inside the App: the main menu and the hidden settings menu, which is for simulation and development only.

4.2.3 EndUser App

The core of the endUser App is collocated in the lib folder. You notice three main folders: **model**, **view**, **controller**.

- **Model**: contains class structures.
- **View**: contains all the application graphics (e.g., various screens).

- **Controller**: contains all the classes related to the handling of HTTP requests and answers.

Chapter 5

Testing

The software testing procedure has progressed with a main emphasis on unit testing. Below is displayed a brief overview of the key evaluations carried out. Due to limited time, the assessment stage has been centered on the server aspect. No unit assessments were performed on the applications, as they are lightweight clients and a majority of the reasoning resides within the server.

5.1 Unit Testing on emspHandler

Insert payment method

- Valid insertion of a new payment method with correct data.
- Attempt to insert a new payment method which has an invalid username.
- Attempt to enter a new payment method that has data that already exists.

Insert booking

- Valid insertion of a new booking with correct data.
- Attempt to insert a new booking which has timeslots and date overlapping.
- Attempt to insert a new booking which has an invalid username.
- Attempt to insert a new booking in a charging station with an id that doesn't exist.
- Attempt to insert a new booking in a charging station where that type of charging socket does not exist.

Search for nearby charging station

- Attempt to insert correct latitude and longititude.
- Attempt to insert invalid or not existing latitude and longititude.

Get personal information

- Valid fetch of the personal information of an existing user.
- Attempt to fetch the the personal information of a not existing user.

Get booking data

- Valid fetch of the booking data thanks to charging station id and socket type.
- Attempt to fetch the booking data with invalid charging station id and valid socket type.
- Attempt to fetch the booking data with valid charging station id and invalid socket type.
- Attempt to fetch the booking data with invalid charging station id and invalid socket type.
- Valid fetch of the booking data thanks to user id.
- Attempt to fetch the booking data with invalid user id.

Delete booking data

- Valid deletion of booking thanks to booking id.
- Attempt to delete a booking with invalid booking id.

Delete payment method

- Valid deletion of payment method thanks to user id and card number.
- Attempt to delete a payment method with valid user id and invalid card number.

Update user password

- Valid update of the user's password.
- Attempt to update the password with an invalid old password and valid user.
- Attempt to update the password with a valid old password and invalid user.
- Attempt to update the password with an invalid old password and invalid user.

Update user email

- Valid update of the user's email.
- Attempt to update the password with an invalid user.

Update user name and surname

- Valid update of the user's name and surname.
- Attempt to update name and surname with an invalid user.

5.2 Unit Testing on cpmsHandler

Get charging station data

- Valid fetch of the charging station data thanks to CPO id.

Get DSO data

- Valid fetch of the DSO data.

Update CPO data

- Valid update of the CPO desired data.
- Attempt to update CPO data with 'nothashed' password.

Insert charging station

- Valid insert of a new charging station.
- Attempt to insert a new charging station with invalid CPO id.

Update dso contract

- Valid update DSO contract.

Update charging mode

- Valid update of charging mode.
- Attempt to update charging mode with an invalid charging station id and valid mode.

Update battery percentage

- Valid update of battery percentage.
- Attempt to update charging mode with an invalid charging station id and valid percentage.

5.3 webSocketHandler testing

This module has been tested using **Postman** which supports a beta functionality to test the WebSocket protocol by emitting and receiving events. In particular, **Postman** supports the Socket.io implementation.

Chapter 6

Installation Instructions

6.1 Server

Here we will illustrate the procedures to follow to deploy the server in a Windows system.

6.1.1 MySQL

1. **Download the MySQL installer**: Go to the MySQL website and download the MySQL installer for your operating system.
2. **Run the installer**: Double-click the downloaded file to start the installation process. Follow the on-screen instructions to complete the installation process.
3. **Choose a setup type**: Select the "Custom" setup type to have more control over the installation process.
4. **Configure the MySQL server**: Set the configuration options as per your requirements. You can choose to install the MySQL as a Windows service or configure the port number that the MySQL server will listen on.
5. **Create a root user**: During the installation process, you will be prompted to create a root user. This is the administrative user for your MySQL server and will have full access to all databases.
6. **Complete the installation**: Finish the installation process and start the MySQL server.
7. **Verify the installation**: To verify that the installation was successful, open the command line interface and run the command "mysql -u root -p". If the installation was successful, you should be able to connect to the MySQL server and run SQL commands.

For sake of simplicity, we recommend to install **XAMPP** software which automatically installs all the essential to run the database engine.

Database Setup

In the *Code/DBMS* folder there is the file .sql to import into your database. To set the credentials you will need to go to the *Code/application-server/index.js*.

6.1.2 NodeJS

To install NodeJS follow the following instruction:

1. **Check the system requirements**: Node.js requires a minimum of 64-bit version of either Windows, macOS, or Linux.
2. **Download the latest version of Node.js**: Visit the official website (https://nodejs.org) and click on the "Download" button to download the latest version of Node.js.
3. **Install Node.js**: Double-click the downloaded executable file and follow the instructions on the screen to install Node.js. On macOS and Linux, you may need to use the terminal to install Node.js.
4. **Verify the installation**: Open the terminal (on macOS and Linux) or the Command Prompt (on Windows) and run the following command: node -v. This will display the version of Node.js that you have installed.
5. **Update the PATH environment variable**: To run Node.js from any location on your system, you need to update the PATH environment variable.

6.1.3 Application server setup

1. Go to *Code/application_server*
2. Open Command Prompt there
3. Run node index.js
4. Now the server is running on *http:///localhost:3000*: if you want to change it, you have to go at *Code/application_server/index.js* and change the default port.

6.2 Client

Here we will illustrate the instruction to install all the three clients (CPO App, EndUser App and Charging socket app).

6.2.1 EndUser App

1. **Download the Flutter SDK**: Visit the Flutter website (flutter.dev) and download the latest version of the Flutter SDK. Extract the ZIP file to a location of your choice.
2. **Set the PATH environment variable**: Add the Flutter SDK's bin directory to your PATH environment variable. This will allow you to run the Flutter command-line tools from any terminal window.
3. **Verify the installation**: Open a terminal window and run the following command: "flutter doctor." This will check if there are any dependencies missing on your system and provide instructions on how to install them.

Then to run a Flutter Project:

1. Navigate to the project directory *Code/end_user_app*
2. Run the project: Run the following command: "flutter run –no-sound-null-safety". This will compile and run your Flutter app on an emulator or a connected physical device.

NOTE:

- Before you run the project, make sure you have an emulator or a connected physical device. If you don't have one, you can create an emulator in the Android or iOS emulator.
- In case of any error go to *Code/end_user_app/.vscode/launch.json* and add this: *"args": ["–no-sound-null-safety"]*
- We discovered that on a specific smartphone you may have some issues in pressing some buttons.

CONFIGURATION NOTE:

- We also provide the apk file inside the folder *APKs/eMallUser.apk*
- The first time you enter in the application, will be required to setup the IP address in which application server is running. Every time you logout from the user settings, the configuration of the address above will be required.

6.2.2 CPO App

To try the CPO App you should follow these steps:

1. Install Node.js and npm (Node Package Manager) if you haven't already (*https://nodejs.org/it/download/*).
2. Open your terminal and navigate to the project directory (*Code/cpo_app/*).
3. Run the command *npm install* to install all the dependencies.
4. Run npm to start the development (*npm start*) server and launch the project in the browser.
5. Now wait for the project to be loaded, then you should navigate to *http://localhost:3000*.

NOTES: The React App can be merged into the Node.Js application server, but once deployed the code and the configurations are no longer available, so for this reason we decided to leave it in development mode (For the moment).

CONFIGURATION NOTES: The HTTP APIs address and the WebSocket APIs address can be modified from the Config.js file in the src folder in the project.

6.2.3 Charging socket App

The APK file is stored *code/APKs/ChargingSocket.apk*. The code of the application is stored in *code/ChargingSocket*, you can open the project with AndroidStudio (*https://developer.android.com/*).

Chapter 7

Effort Spent

7.1 Team discussions

Topic	Hours
General discussion	3h
Document writing	8h
General system testing	6h

7.2 Balestrieri Niccolò

Topic	Hours
Application server development	45h
Application server testing	15h

7.3 Bertogalli Andrea

Topic	Hours
CPO App	50h
Charging socket App	10h

7.4 Tombini Nicolò

Topic	Hours
End user App	60h

Chapter 8

References

- *React.JS Docs*
- *Flutter Docs*
- *Stackoverflow.com*
- *MUI Docs*
- *Node.JS Docs*