Software Engineering 2 - Prof. Di Nitto Elisabetta
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

eMall - e-Mobility for All

ITD
Implementation and Testing Deliverable

February 5, 2023

Giovanni De Lucia (10700658)
Lorenzo Battiston (10618906)
Matteo Salvatore Currò (10940719)

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

This document outlines the implementation and testing procedures that have been followed to develop a functioning prototype of the service described in the ”Requirements Analysis and Specification Document” and ”Design Document”. It is intended to serve as a reference for the development team, detailing the software, frameworks, and programming languages chosen.

1.2 Definitions, Acronyms and Abbreviations

1.2.1 Definitions

- EV Driver - Electric Vehicle Driver, people or entities who own an EV car and want to use the system for their charging needs. In this document they can be also referred to as ”users”.
- EVCP - Electric Vehicle Charging Pool, is a station with multiple CPs
- CP - a synonym of EVSE - is a single charging column with multiple connectors
- Connector (Socket) - charging socket that can be of different types (e.g. CCS2, Type2)
- Rate - the rate that the CPO decides to set for the CPs it manages. It contains a fixed part for parking and a variable part per kWh. Usually the rates are associated with a certain power (kW)

1.2.2 Acronyms

RASD	Requirement Analysis and Specification Document
DD	Design Document
API	Application Programming Interface
CPO	Charging Point Operator
DSO	Distribution System Operator

1.2.3 Abbreviations

$\mathbf{R}_x$	x-Functional Requirement

1.3 Revision history

Revised on	Version	Description
31-Jan-2023	1.0	Initial Release of the document

1.4 Reference Documents

- Requirement Analysis and Specification Document (referred to as “RASD” in the document)
- Design Document (referred to as DD in the document)
- Assignment document A.Y. 2022/2023 (”Requirement Engineering and Design Project: goal, schedule and rules”)

1.5 Document Structure

This document is composed of six sections:

- Introduction

- Product Functions: presents the implemented and discarded functions of the prototype.
- Development Frameworks: presents the adopted programming languages and frameworks, justifying each choice.
- Code Structure: presents the structure of the code
- Testing: explains how and what has been tested
- Installation Guide: provides explanations on how to run, test and build the prototype

2 Product functions

In this section, we present the implemented and discarded functional requirements of the prototype.

2.1 CPO functions

- **R1** The system must allow unregistered CPO to register an account
Implemented: Yes, in the form of a registration page
- **R2** The system must allow registered CPO to login
Implemented: Yes, in the form of a login page
- **R3** The system must allow authenticated CPOs making a special offer on their CPs prices
Implemented: Yes, it is possible to make a special offer on the price of an entire EVCP in the Rate tab of the CPO dashboard
- **R4** The system must allow authenticated CPOs monitoring the charging process to infer when the battery is full
Implemented: No, it is not possible to monitor the charging process. Although we have implemented a simple socket simulator, it is not possible because it would need sensors to monitor the charging process
- **R5** The system must allow authenticated CPOs retrieving the amount of energy available in their EVCPs batteries
Implemented: No, it is not possible to retrieve the amount of energy available in the batteries of the EVCPs. We made a mocked 'Energy API' that returns a random value given a key, but it is not possible to retrieve the actual value of the batteries
- **R6** The system must allow authenticated CPOs retrieving the number of vehicle being charged in their EVCPs and for each vehicle the amount of absorbed power
Implemented: +/-, it is possible to retrieve the number of vehicle being charged in their EVCPs but for the same reasons as R4, it is not possible to retrieve the amount of energy being absorbed
- **R7** The system must allow authenticated CPOs retrieving the remaining charge time for each connected vehicle
Implemented: +/-, it is possible to retrieve the end of a reservation but not an estimation of the remaining charge time, for the same reasons as R4
- **R8** The system must allow authenticated CPOs retrieving details on active and historical reservations on their EVCPs
Implemented: Yes, in the reservations tab of the CPO dashboard
- **R9** The system must allow authenticated CPOs acquiring information from the DSOs about the current price of energy
Implemented: Yes, via a mocked 'DSO API' in the Energy tab of the CPO dashboard
- **R10** The system must allow authenticated CPOs deciding from which DSO to acquire energy from
Implemented: Yes, via a mocked 'DSO API' in the Energy tab of the CPO dashboard
- **R11** The system must dynamically decide where to get energy for charging (electrical grid, battery or a mixture)
Implemented: No
- **R12** The system must allow authenticated CPOs statically deciding where to get energy for charging (electrical grid, battery or a mixture)
Implemented: +/-, via the Energy API and the battery key it is possible to set an operating mode for the battery
- **R13** The system must allow authenticated CPOs adding, modifying and deleting CPs
Implemented: Yes, in the Charging Points tab of the CPO dashboard it is possible to create EVCP, adding CP to them and then sockets to the CP
- **R14** The system must allow authenticated CPOs changing availability status of their CPs
Implemented: No

2.2 Driver functions

- **R15** The system must allow unregistered users to register an account
  Implemented: Yes, in the form of a registration page
- **R16** The system must allow registered users to login
  Implemented: Yes, in the form of a login page
- **R17** The system must allow authenticated users to personalize their experience by providing information of their EV
  Implemented: No, it was not requested, and it would have slowed down the development process
- **R18** The system must allow users to search for CPs in the map
  Implemented: Yes, in the map tab of the driver
- **R19** The system must show to the users CPs nearby their current position
  Implemented: Yes, in the map tab of the driver
- **R20** The system must allow retrieving details on a given CP regarding connector types supported and cost of the charge
  Implemented: Yes, in the map tab of the driver by clicking on a marker representing an EVCP
- **R21** The system must allow authenticated user to book a CP for a certain time interval
  Implemented: Yes, in the map tab of the driver by clicking on a marker representing an EVCP and then on the book button
- **R22** The system must allow booking a CP if and only if it is free for the specified time interval
  Implemented: Yes, via a check on the availability of the EVCP in the backend
- **R23** The system must notify users when the charging shift is about to start
  Implemented: No, it was not requested, and it would have slowed down the development process
- **R24** The system must allow authenticated users to start the charge
  Implemented: Yes, in the reservation tab of the driver
- **R25** The system must suggest users when and where to charge based on daily schedule, special offers and availability
  Implemented: No, it was not requested
- **R26** The system must allow authenticated users to monitor the charging status
  Implemented: +/-, it is possible to monitor the charging status but not the actual energy being absorbed, for the same reasons as R4
- **R27** The system must notify authenticated users when the charging process is completed
  Implemented: Yes, via Firebase Cloud Messaging and a cron job that checks the status of the reservation every 5 minutes
- **R28** The system must allow authenticated users to pay for the charge
  Implemented: No, it was not requested
- **R29** The system must allow authenticated users to delete a reservation
  Implemented: No, it could have been implemented, but it was not requested, and it would have slowed down the development process
- **R30** The system must allow authenticated users to view historical reservations
  Implemented: Yes, in the reservation tab of the driver

3 Development frameworks

In this section, we will present the adopted programming languages and frameworks, justifying each choice.

3.1 Adopted programming languages

3.1.1 JavaScript

For both the backend and the frontend, we have chosen to use the programming language **JavaScript**. This choice was made because it is a language that is widely used in the industry, and it is also the language that we are most familiar with. There are many advantages to using JavaScript, such as the fact that it is a multi-paradigm language, and it is also very easy to learn. It is a good language to design and implement prototypes fastly, and it is also a good language to use for small projects.

Pros

- Widely used in the industry
- Easy to learn
- Multi-paradigm language
- Good for small projects
- Good for prototyping

Cons

- Not a good language for large projects
- Not a good language for performance-intensive applications
- Not type-safe

Probably, the biggest disadvantage of JavaScript is that it is not a type-safe language and the use of TypeScript is a better choice for large projects. The reasons for behind the choice of not using TypeScript is that it is more verbose and this lead to a slower development process.

3.2 Framework and Libraries

3.2.1 Node.js

Node.js is a runtime environment for JavaScript that allows us to run JavaScript code outside of a browser. So it is used to run JavaScript on the server-side. It is a good choice for our project because it is a very popular framework and it is served by a large community of developers that provides useful packages and libraries on `https://www.npmjs.com`.

3.2.2 Vite

Vite is a build tool that is used to bundle our code and to serve it to the browser. It is a good choice for our project because it is very faster than alternatives like Webpack. It is used also to create a development server that allows us to see the changes in real-time thanks to its HMR (Hot Module Replacement) feature.

3.2.3 React

React is a widely used JavaScript library for building user interfaces. It is a good choice for developing a prototype because it allows us to create an SPA (single-page application) that is very fast and responsive. React is used with modularity in mind, so it is easy to create components that can be reused in different parts of the application, leading to clean and maintainable code.

3.2.4 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that allows us to create a responsive and mobile-first design. It permits the fast style prototyping of the application. A drawback of Tailwind CSS is that it is not a framework, so it does not provide any components that can be reused in the application. But used along with React, it is possible to create styled and reusable components.

3.2.5 Jest

Jest is a testing framework that allows us to test our code. It is used to test the backend and the frontend. We used it to test the backend main functionalities. It can also generate useful reports with line and branch coverage.

3.3 API

3.3.1 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that allows us to send push notifications to our users. We use it to notify the driver when a charge has been completed.

3.3.2 Other APIs

Other APIs has been mocked to simulate the real behavior of the system. For example, the API that allows the CPOs to retrieve the list of available DSOs, or the SMS and email API. This choice was made because some of these APIs are not available for free, others are not available at all.

3.4 Other tools

3.4.1 Postman

Postman is a tool that allows us to test our backend APIs without having to write any client code. It can be also used to generate the documentation of the APIs.

3.4.2 Docker

Docker is a tool that allows us to create containers that can be used to run our application in a production environment. It is used to create a container that runs the backend and the frontend of the application. This choice was made because it simplifies the deployment of the application but also because it hides inconsistencies between different operating systems.

3.4.3 Jsdoc

Jsdoc is a tool that allows us to generate the documentation of our code. It is used to generate the documentation of the backend of the application.

4 Source Code Structure

In this section we describe the structure of the source code of the application. To be more clear, we will describe only the most important parts of the code and the relevant files.

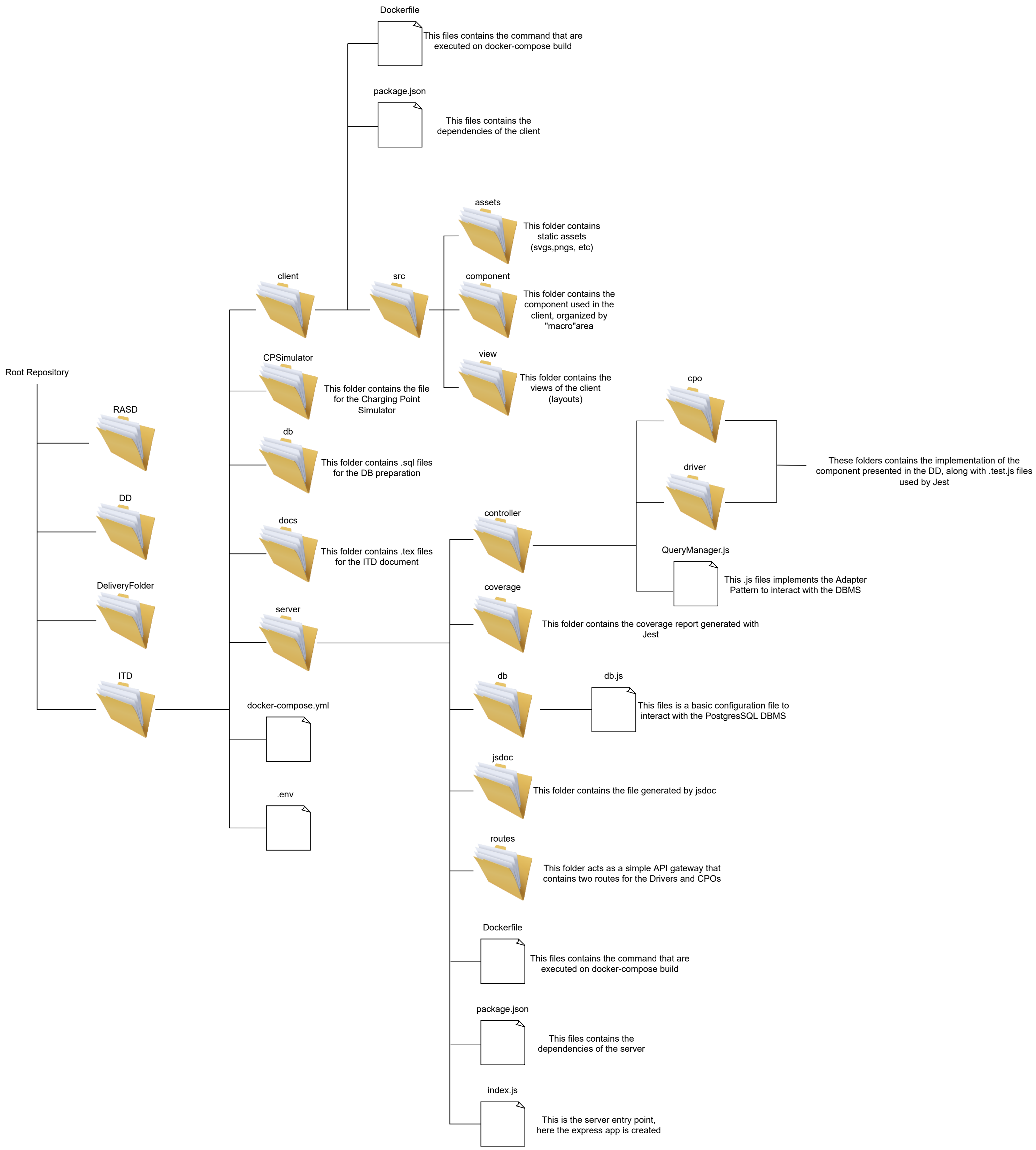

Figure 1: Source code structure of the application.

5 Testing

5.1 Test cases and outcomes

6 Installation Guide

For the installation guide refers to this wiki page: `https://github.com/gio-del/BattistonDeLuciaCurro-swe2/wiki/Installation-Guide`