ATD

Software Engineering Project
A.Y. 2022-2023

Marco Ronzani, Alessandro Sassi

December 2022 - February 2023

Contents

1 Introduction

1.1 Purpose

This Acceptance Test Document is meant to validate and test the implemented prototype of the eMall software developed by another team in the Software Engineering 2 course at Politecnico di Milano.

1.2 Scope

The eMall project which is going to be tested is meant to offer users the possibility to find Charging Stations for their electric vehicles, book charges at those stations and manage them, all from a single platform that is meant to dialogue with the software systems of the various Charge Point Operators in order to present to above functionalities.

The authors of the project which will be discussed are:

- Alireza Yahyanejad – 10886993
- Mohammad Hosein Behzadifard – 10880732
- Alireza Azadi - 10888648

Here is the GitHub repository containing their project:
https://github.com/alirezaazadi/AzadiBehzadifardYahyanezhad

The documents taken into account for the project's testing were the RASD, DD and ITD documents found in the folder "/DeliveryFolder" within the GitHub repository (link).

2 Installation Setup

The team provided some installation instruction in their ITD document. These were the instructions that we tried to follow at the beginning, however we found that they were fairly incomplete, since the Docker container could not be built without the appropriate ".env" files.
After this, we followed the instructions provided in the README in the folder of each subsystem, but in the case of the CPMS the folder where the ".env" file had to be placed was not found and the instructions did not specify that it had to be created and where to do so.
We can conclude that the installation instructions were the bare minimum with only two loosely described commands presented in the IT document which, by themselves, would not have let us run the project. Furthermore, the additional missing instructions were present only in the "README.md" files inside the subfolder of each component of the software.
Since Docker files were used to render the installation seamless, it is understandable that the installation documentation was kept short, nevertheless it would have been much handier to have all the instructions in one place and in clear order of execution rather than scattered in different folders without specifying a clear order.
Other instructions that were not provided were related to the default user for the CPMS (without which we could not get access to the CPMS frontend) and the Django command `createsuperuser` to initialize the default CPMS account.

2.1 Steps Followed

The following are the steps we followed to run the projects:

1. **Cloning of the GitHub repository** on our local machines.
2. **Creating the ".env" files** following the instruction in the README files (and, for the CPMS backend, the Docker error for the correct file path), with the required configuration for the CPMS and eMSP backends. The files were created in the *../cpms/settings/environments* and *../emsp/settings/environments* folders, respectively.
3. **Installing Docker**, since our machines did not have Docker already installed. However, this step was not mentioned during the installation instructions, hence for additional clarity in the ITD we suggest it to be at least cited as a prerequisite.
4. **Composing the Docker containers** for each of the four components (CPMS and eMSP backend, CPMS and eMSP frontend). The docker command used to compose their container was run in their respective folders, obtaining 4 running containers. The commands used were:
   - **eMSP backend**: `docker compose -f docker-compose.yml -f docker-compose.prod.yml --env-file cpms/settings/environments/.env up -d`
   - **CPMS backend**: `docker compose -f docker-compose.yml -f docker-compose.prod.yml --env-file emsp/settings/environments/.env up -d`
   - **CPMS frontend**: `docker compose up -d --build`
   - **eMSP frontend**: `docker compose up -d --build`
5. **Database Migration**, required by the eMSP backend module, performed by starting the bash CLI interface of its Docker container and running the command provided in *../backend/emsp/README.md*.
   - **Starting the CLI**: `docker exec -it emsp_backend /bin/bash`
   - **Migrating the DB**: `python manage.py migrate`

2.2 Issues Encountered

Two different issues were encountered while running the `docker compose` on our two machines, one for each:

- **Docker runtime crashing**, while running `docker compose` on any of the four components of the project with a fresh installation of Docker, the command continued to end abruptly, without any debug information.
  Solution: None currently.
- **Docker runtime hanging**, after running `docker compose` on the two frontend components, for both the Docker process became stuck on `[2/7] RUN apk add python3 make g++`, and did not recover even after multiple hours (Note: the machine hardware is current-gen and offers high performance). **Solution**: After discussing the issue with the group who developed the prototype, they suggested running the `docker compose` commands one at a time, never concurrently, and they did complete successfully.
- **manage.py not runnable**, the `manage.py` program is the core for testing and generating fake data, however it could not be run withing the provided container. **Solution**: We found out that the problem was the use of the wrong file formatting with respect to the container, by converting the file from dos to unix format, it worked. For more details see https://stackoverflow.com/questions/19425857/env-python-r-no-such-file-or-directory.
- **CPMS database not connecting to the backend**, after a restart of the docker containers, we were not able anymore to make the CPMS's backend work due to a connection issue with its database. **Solution**: After discussing the issue with the group who developed the prototype and after they conducted tests on their machines, we were unable to solve the issue and settled on running only the prototype's frontends on our machine, having them connect to backends which were hosted by the other team.

3 Acceptance Tests

3.1 Provided Functionalities

Within the provided ITD document there was a list of implemented functionalities, but non of them reported on which requirements it satisfied, consequently we settled on testing every requirement, marking those which were clearly not satisfied due to their respective functionalities being absent from the list present in the ITD document as "Not implemented".
Those are the reported implemented functionalities:

1. **EMSP**
    - Sign up and Sign in
    - Visit and Edit profile (Current user profile)
    - Show a list of charging stations
        - Sort them based on the distance from the current user location
        - Sort them based on the offer (discount) and price
        - Retrieve details of a Charing Stations and information about its socket
    - Book a Charging Socket for a Specific time frame
    - Access to the list of booked items:
        - Filter booking history based on the status
        - Order based on their date, price, and status
        - Get the payment QR code for a successful booking
2. **CPMS**
    - Login to the panel
    - Retrieve a list of Charging stations
        - Sort them based on the location
        - Receive its statuses:
            * External Status: number of charging sockets available, their type such as slow/fast/rapid, their cost, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed.
            * Internal Status: the amount of energy available in its batteries, if any, 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.
        - Setup its power source: It can be DSO, Battery, or Mixed.
    - Manage DSOs
        - Retrieve a list of DSOs
        - Set a DSO as the main DSO for the system to acquire energy
        - Filter them based on their availability
        - Sort them by price

- Managing connected vehicles to the charging sockets: start charging a vehicle according to the amount of power supplied by the socket, and monitor. the charging process to infer when the battery is full

3.2 Considered Goals

The project was intended to satisfy the following goals presented in the associated RASD document:

Goal	Description
G1	eMSP allows end user to know about the charging stations nearby, their cost, any special offer they have
G2	eMSP allows end user to book a charge in a specific charging station for a certain timeframe
G3	eMSP allows end user to start the charging process at a certain station
G4	eMSP allows end user to be notified when the charging process is finished
G5	eMSP allows end user to pay for the obtained service
G6	eMSP allows end user to sort the available stations
G7	CPMS allows CPO to know the location of a charging station
G8	CPMS allows CPO to know the external status of a charging station
G9	CPMS allows CPO to decide from which DSO to acquire energy
G10	CPMS informs CPO to dynamically decide where to get energy for charging
G11	CPMS allows CPO to acquire by the DSOs information about the current price of energy
G12	CPMS allows CPO to know the internal status of a charging station
G13	CPMS allows CPO viewing the charging process to infer when the battery is full
G14	CPMS allows CPO to know the time to start charging a vehicle according to the amount of power supplied by the socket
G15	CPMS allows CPO sets current price of energy for paying of services

3.3 Considered Requirements

The project was intended to satisfy the following requirements, presented in the associated RASD document:

Id	Requirement
R1	The eMSP must allow an unregistered end user to register
R2	After an end user fills all the blanks on registration page correctly, the system must send an email to her/him, in order to confirm her/his email
R3	The eMSP must allow a logged-out end user to login
R4	The CPMS must allow CPO to login
R5	The CPMS must allow CPO to view the list of charging stations with their statuses
R6	The eMSP must notify the user when the charging process is finished through notification
R7	The eMSP must access to the GPS of the end user smartphone for suggesting charging stations
R8	The CPMS must allow a logged-out CPO to login
R9	The eMSP must notify the user when there is an error during charging process through notification
R10	The eMSP must allow end user to view her/his information
R11	The eMSP must allow end user to enter her/his information
R12	The CPMS can connect to several eMSPs through API
R13	The CPMS must connect to each charging points
R14	The eMSP must allow end user to pay for the service
R15	The eMSP must create QR code if end user wants to pay by QR code
R16	The eMSP must allow end user to book a charging point
R17	The eMSP must access to the end user's calendar
R18	The eMSP must suggest charging points depending on the status of the battery
R19	The CPMS must show external status to the CPO
R20	The CPMS must show internal status to the CPO
R21	The CPMS must show locations of charging stations
R22	The CPMS must allow CPO to insert the current price for services
R23	The eMSP must show the receipt after end user paid for the services
R24	When the car is charging, the eMSP must show the exact status of the battery to the end user
R25	When the end user is on the main page, eMSP must shows the list of all charging stations in order
R26	eMSP must shows the prices of each charging stations to the end user in main page
R27	eMSP must allow end user to select the way of payment
R28	eMSP must show the ways of payments to the end user
R29	After an end user inserted all the personal information related to her/him, the eMSP must allow her/him to confirm
R30	eMSP must allow the end user to edit information about her/his credit card

Id	Requirement
R31	After an end user inserted all the information related to her/him credit card, the eMSP must allow her/him to confirm
R32	eMSP must allow end user to cancel the time she/he booked
R33	The CPMS must show available DSOs to the CPO
R34	When end user wants to see available charging stations, the eMSP must show special offers that CPO considered for the end user
R35	After an end user books a charging point, the eMSP must send a notification to her/him to inform her/him
R36	eMSP must allow the end user to edit her/his information
R37	eMSP must allow the end user to see the status of her/his car battery
R38	The CPMS must allow CPO to view the current price of energy through DSOs
R39	The eMSP must show available charging stations to the user
R40	The eMSP must show the charging stations' status to the user
R41	The CPMS must allow CPO to select DSOs to acquire energy
R42	The CPMS must show how a vehicle is charging to the CPO
R43	The CPMS must allow the CPO to insert offers for end users

3.4 EMSP Tests

As per standard procedures of acceptance testing, we decided to test the project on a per-requirement basis, and since all the implemented functions were meant to satisfy the requirements, consequently every functionality was tested as well.

3.4.1 Signup Test - R1

- **Test procedure**:
  - From the login page, click on the "Sign up!" button
  - Fill the Sign Up form with Name, Surname, Email, Password
  - Click on the "Sign Up" button
- **Acceptance criterion**: A "registration successful" popup appears after the registration.
- **Result**: Pass

3.4.2 Confirmation Email Test - R2

- **Test procedure**:
  - From the login page, click on the "Sign up!" button
  - Fill the Sign Up form with Name, Surname, Email, Password
  - Click on the "Sign Up" button
- **Acceptance criterion**: The registration email has been received.
- **Result**: Fail

3.4.3 Customer Login Test - R3

- **Test procedure**:
  - From the login page, fill the Login form with Email and Password
  - Click on the "Login" button
- **Acceptance criterion**: The login is successful and the user is redirected to the Home page.
- **Result**: Pass

3.4.4 Charging process finished notification - R6

Not implemented.

3.4.5 Location-based CSs search - R7

- **Test procedure**:
  - From the home page, enter a start and end date and time and price range
  - Click on the "continue" button
  - The list of CSs is sorted according to distance to the user
- **Acceptance criterion**: The list of CSs is sorted according to distance to the user
- **Result**: Fail (not implemented)

3.4.6 Charging process error notification - R9

Not implemented.

3.4.7 Show User Personal Information - R10

- **Test procedure**:
  - From the home page, click on the Profile icon
- **Acceptance criterion**: The Profile page with the user personal information appears
- **Result**: Pass

3.4.8 Insert User Personal Information - R11

Variation of the already performed test for R1.

- **Test procedure**:
  - From the login page, click on the "Sign up!" button
  - Fill the Sign Up form with Name, Surname, Email, Password
  - Click on the "Sign Up" button
  - Click "Login" after the redirection to the login page
- **Acceptance criterion**: The inserted user information is correctly displayed on the home page
- **Result**: Pass

3.4.9 User Payment - R14

Not required for the implementation part of the project.

3.4.10 Pay via QR Code - R15

Not required for the implementation part of the project.

3.4.11 CS Booking - R16

- **Test procedure**:
  - From the home page, enter a start and end date and time and price range
  - Click on the "continue" button
  - Click on one of the available CSs
  - Click on "Book" for one of the CS's sockets
- **Acceptance criterion**: A popup confirming the booking appears and the bookings page is shown
- **Result**: Pass

3.4.12 Calendar Access - R17

Not implemented.

3.4.13 Battery State-Based Suggestion - R18

Not implemented.

3.4.14 Signup Test - R23

Not required for the implementation part of the project.

3.4.15 Battery Reporting - R24, R37

Could not be done through the EMSP's frontend, since no sufficient API documentation was provided, the requirement has been deemed unsatisfied and could not be tested.

3.4.16 Home Page CS List with Prices - R25, R26, R39

- **Test procedure**:
  - From the home page, enter a start and end date and time and price range
  - Click on the "continue" button
  - Click on one of the available CSs
- **Acceptance criterion**: The expanded view of CSs appears with all the details
- **Result**: Pass

3.4.17 Select, Show, Update and Confirm User Payment Methods - R27, R28, R30, R31

Not required for the implementation part of the project.

3.4.18 Confirmation After Personal Information Change - R29

- **Test procedure**:
  - From the Profile page, update all user information (user name and surname, email, password)
  - Click "Update"
- **Acceptance criterion**: A mean of confirming the update appears
- **Result**: Fail (nothing appears, only a "change successful" notification)

3.4.19 Booking Cancellation - R32

- **Test procedure**:
  - From the home page, open the navigation menu in the top right
  - Click on the "Booking History" button
  - Click on a booking in the list to select it
  - Deletion of the selected booking from the ones present in the page
- **Acceptance criterion**: A "booking deleted" confirmation message appears
- **Result**: Fail (not implemented)

3.4.20 Show Special Offers - R34

- **Test procedure**:
  - From the home page, enter a start and end date and time and price range
  - Click on the "continue" button
  - Click on one of the available CSs
- **Acceptance criterion**: The expanded view of CSs clearly shows that a discount is applied, the entity of the applied discount or both the non-discounted and discounted prices
- **Result**: Fail (no clear element in the UI indicates the presence of a discount)

3.4.21 Booking Notification - R35

Not implemented.

3.4.22 Update User Personal Information - R36

- **Test procedure**:
  - From the Profile page, update all user information (user name and surname, email, password)
  - Click "Update"
- **Acceptance criterion**: A popup confirming the successful update appears
- **Result**: Pass

3.4.23 Show CS Status - R40

Not implemented.

3.5 CPMS Tests

3.5.1 CPO Login - R4, R8

Note that there is a repetition in the RASD document, R4 and R8 coincide.

- **Test procedure**:
  - From the login page, fill the Login form with Username and Password
  - Click on the "Login" button
- **Acceptance criterion**: The login is successful and the user is redirected to the Home page.
- **Result**: Pass

3.5.2 View list of CSs - R5

- **Test procedure**:
  - Login performed, Home page reached
  - Select "Charging Stations" from the navigation menu on the left
- **Acceptance criterion**: The list of CSs is presented to the User
- **Result**: Pass

3.5.3 Multiple CPMSs - R12

Could not be tested with the provided project material, since there is no way of concurrently running multiple instance of the CPMS component, their ports end up conflicting. Or even if it was possible, instructions on how to perform the above were not provided.

3.5.4 CPMS CSs Connection - R13

Done under the assumption that having a CSs show up in the CPMS UI implies such CS being connected to the CPMS.

- **Test procedure**:
  - Login performed, Home page reached
  - Select "Charging Stations" from the navigation menu on the left
- **Acceptance criterion**: Every supposedly connected CS is presented to the User
- **Result**: Pass

3.5.5 CS External Status - R19

- **Test procedure**:
  - From the home page, select "Charging Stations" from the navigation menu on the left
  - From the CS list, click on the "see more" button of a CS
- **Acceptance criterion**: The information on the external status of a CS (number of charging sockets available, their type, 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) is displayed

- **Result**: Pass

3.5.6 CS Internal Status - R20

- **Test procedure**:
  - From the home page, select "Charging Stations" from the navigation menu on the left
  - From the CS list, click on the "see more" button of a CS
  - If possible, for a socket with a connected vehicle, click on the "start charge" button
- **Acceptance criterion**: The information on the internal status of a CS (amount of energy available in its batteries, if any, number of vehicles being charged and, for each charging vehicle, amount of power absorbed and time left to the end of the charge) is displayed
- **Result**: Pass

3.5.7 CS Location - R21

- **Test procedure**:
  - From the home page, select "Charging Stations" from the navigation menu on the left
  - From the CS list, click on the "show on map" button of a CS
- **Acceptance criterion**: A Map is loaded, displaying the CS's location
- **Result**: Pass

3.5.8 Prices Update - R22

Not implemented.

3.5.9 Available DSOs - R33

- **Test procedure**:
  - From the home page, select "DSOs" from the navigation menu on the left
- **Acceptance criterion**: The list of all available DSOs is shown
- **Result**: Pass

3.5.10 DSOs Energy Prices - R38

- **Test procedure**:
  - From the home page, select "DSOs" from the navigation menu on the left
- **Acceptance criterion**: The current energy price for each DSO is shown
- **Result**: Pass

3.5.11 DSO Selection - R41

- **Test procedure**:
  - From the home page, select "DSOs" from the navigation menu on the left
  - Click on the "acquire energy" button of a DSO

- **Acceptance criterion**: A notification appears confirming the update to the current DSO
- **Result**: Pass

3.5.12 Show Vehicle Charge Status - R42

- **Test procedure**:
  - From the home page, select "Charging Stations" from the navigation menu on the left
  - From the CS list, click on the "see more" button of a CS
  - If possible, for a socket with a connected vehicle, click on the "start charge" button
- **Acceptance criterion**: The charge status of the vehicle is displayed
- **Result**: Pass

3.5.13 Insert Offers - R43

Not implemented.

3.6 Automated Tests

We tried running the automated tests described in the ITD document associated with the project, those were the result:

- **CPMS backend**

```
python manage.py test
Found 0 test(s).
System check identified no issues (0 silenced).

----------------------------------------------------------------------
Ran 0 tests in 0.000s

OK
```

  We can only deduce that no tests were provided for the cpms's backend.

- **EMSP backend**

```
python manage.py test
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.450s

OK
Destroying test database for alias 'default'...
```

  Only two tests were provided, and correctly succeeded.

3.7 Reporting on Goals

With respect of which requirements were implemented, and which passed their respective tests, we can classify the goals as follows:
(Pleas note that no goals-to-requirements mapping was provided)

Goal	Description
Satisified	G2, G6, G7, G8, G9, G10, G11, G12, G13, G14
Partially satisifed	G1 (missing special offers)
Not satisfied	G3, G4, G5. G15

4 Additional Comments

The provided RASD and DD documents are affected by some repetitions and unclear statements, but overall were usable for the testing procedures. Unfortunately, as already mentioned, the ITD document proved of little use for the testing procedures, lacking first and for most a clear reference to the requirements discussed in the previous documents, with only a quick functionalities list being present, and secondly lacking clear and extensive instructions on how to run the project. Such limitations made the task of testing the project harder, and it is possible that some functionalities were not tested as intended by the team who developed the project as a consequence.

That being said, we have to highly praise the members of the reviewed group for being available to help us setting up their project and solve the issues we encountered. Even if not everything went as planned, they committed themselves to helping us, and thanks to that we were able to complete the testing process.

5 Effort Spent

5.1 Ronzani Marco - mat: 224578

Task	**Time spent**
Installation	$6h$
Testing	$7h$
This document	$3h$
Total	$16h$

5.2 Sassi Alessandro - mat: 220837

Task	**Time spent**
Installation	$7h$
Testing	$0h$ (due to installation issues)
This document	$2h$
Total	$9h$

5.3 References

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