

***“PowerEnjoy”***

*Project Management Document*

**Version 1.0** *(11/12/2016)*

Giorgio Marzorati (876546)

Aniel Rossi (877018)

Andrea Vaghi (877710)

INDEX OF CONTENTS

INTRODUCTION 3

Purpose 3

Scope 3

Reference Documents 5

Document Structure 5

PROJECT SIZE, COST AND EFFORT ESTIMATION 6

Size estimation: function points 6

Internal Logic Files (ILFs) 6

External Logic Files (ELFs) 7

External Inputs (EIs) 7

External Inquiries (EQs) 7

External Outputs (EOs) 7

Overall estimation 8

COST AND EFFORT ESTIMATION: COCOMO II 8

Scale Drivers 8

Cost Drivers 8

Schedule estimation 9

Effort equation 9

Schedule estimation 9

SCHEDULE 10

RESOURCE ALLOCATION 10

RISK MANAGEMENT 11

EFFORT SPENT 12

Giorgio Marzorati 12

Aniel Rossi 12

Andrea Vaghi 12

CHANGELOG 13

# **INTRODUCTION**

## Purpose

This document aims to provide design-level technical details on the architecture of PowerEnjoy, the mobile application of car sharing that we are going to implement.

It is addressed mainly to developers and stakeholders with at least some technical knowledge about architectural design and implementation.

The document will first analize high level components and give an overall description of the architecture. Those components will be then “unpacked” and described more precisely with a top-down approach, as well as the way they interact with each other. The document will analize:

* High level architectures and the identification of the tiers
* Adopted design patterns
* Main components and their interaction
* Runtime behaviour with some UML diagrams & implementation pseudocode

## Scope

PowerEnjoy is a car-sharing service thought for the city of Milan based on a mobile application with a single category of users.

The system allows clients to reserve, or directly identify, and use available electric-powered cars in the area around the client’s GPS position or around an address inserted manually. In case of reservation, if the client does not identify the car within 1 hour from the reservation it is deleted and he is charged with a fee of 1€. Then the car becomes available again to other clients.

In order to use the application, clients have to register to the system, in particular they have to provide an e-mail address, biographical data and a valid driving license. On the other side, the system provides clients a personal PIN, with which they access to the system and are allowed to interact with car.

Clients are charged at the end of every ride and the payment is done with one of the supported payment methods that must be specified during the registration process and it can be modified in every moment.

In addition, the system defines different discounts and overcharges for every ride, as a result of particular client’s behaviours (more informations are included in the Glossary session).

The system has the purpose of providing an efficient and environment-friendly alternative to public transportation to people who don’t have to cover long-distance travels and don’t want to (or cannot) use personal vehicles.

**Definitions, Acronyms, Abbreviations**

1. **API**: Application Program Interface, it exposes a set of public methods used to make two different systems communicating with each other
2. **BLL**: Business Logic Layer, is the central layer of the three-tier architecture. It’s represented by the central server and its components
3. **CL**: Client Layer, is the first level of the three-tier architecture. It’s represented by the Client, Cars and the Assistance Team
4. **DBMS**: Database Management System
5. **DD**: Design Document
6. **DL**: Data Layer, is the third and last level of the three-tier architecture. It’s represented by the DBMS
7. **E-Mail Gateway**: it’a service used by the central server to send emails to clients
8. **JDBC Driver**: is a native Java connector used by the central server to comunicate with any relational DBMS
9. **HTTP**: HyperText Transfer Protocol, application-level protocol for exchanging information on the web, in a client-server arhitecture
10. **MQTT**: MQ Telemetry Transport, application-level protocol with a public-subscribe pattern
11. **RASD**: Requirements Analisys and Specification Document
12. **REST**: REpresentational State Transfer
13. **RESTful**: Web Service based on a REST architecture
14. **Top-Down approach**: is a descriptive model based on an iterative analisys, starting from a more generic and high level rappresentation of an architecture. At every iteration, the model is divided into more specific components which are also analized and divided.
15. **UML**: Unified Modeling Language, a graphic language used to represent different aspects (static, dynamic, architectural, behavioural..) of a specific software.
16. **URL**: Uniform Resource Locator
17. **UX**: User Experience

## Reference Documents

* Assignments+AA+2016-2017.pdf
* Examples documents:
  + Sample Design Deliverable Discussed on Nov. 2

## Document Structure

**Introduction**: the section explains the purpose of the document and underlines the main differents between DD and the RASD.

**Architecture Design**:

1. Overview: shows the overall architecture from an high level point of view.

2. Component view: provides an high level diagram and a more detailed one, explaining the purpose of the most important components and the interaction between tiers

3. Deploying view: this section shows the components that must be

deployed to have the application running correctly.

4. Runtime view: sequence diagrams are represented in this section to

show the course of the different tasks of our application

5. Component interfaces: this section presents the most important interfaces between components

6. Selected architectural styles and patterns: this section explain the

architectural choices taken during the creation of the application

**Algorithms Design**: this section describes the most critical parts via some algorithms written in pseudocode.

**User Interface Design**: this section presents the user experience explained through an UX diagram.

**Requirements Traceability**: this section aims to explain how goals identified in the RASD are satisfied with the usage of design components.

# PROJECT SIZE, COST AND EFFORT ESTIMATION

In the next section we will at first present a valuation of the expected size of the PowerEnjoy project (expressed in SLOC), then an estimate of the cost and the required effort, using the following approaches:

* Function Points for the size estimation, based on each functionality that the business logic of the application has to offer and the corresponding lines of code to be written in order to correctly implement it;
* COCOMO II approach for the cost and the required effort, taking into account that we find ourselves into a Post-Architecture case of analysis

## Size estimation: function points

### Internal Logic Files (ILFs)

In this paragraph we will describe the Internal Logic Files on which the PowerEnjoy data structure is based on in order to be able to implement all the functionalities it offers. We will focus in particular in the description of their complexity, to justify the amount of FPs we decided to apply to every single ILF.

ILF Rating (Low = 7 FPs, Avg = 10 FPs, High = 15 FPs):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data Elements** | | |
| **Record Elements** | 1-19 | 20-50 | 51+ |
| 1 | Low | Low | Avg |
| 2-5 | Low | Avg | High |
| 6+ | Avg | High | High |

1) Client

Attributes:

* Driving License id
* Email
* PIN
* Name
* Surname
* Card Circuit
* Card Expiration Date
* Card Number
* Card CVV

### External Interface Files (EIFs)

In this section we will explain the interaction between PowerEnjoy and some sets of data provided by other independent applications/providers. In particular our system relies on two main external services, the *Mapping Service* and the *Payment Service Provider*. The information retrieved through the interaction between the system and these components sometimes need a certain amount of analysis and processing in order to be useful in our system. There may be situations in which the complexity of some of those interactions needs to be considered in a proper way, taking into account the dimension of the data exchanged and the amount of business logic required to process it. The list of the interactions is the following:

**Mapping Service:**

* City Map Retrieval:

This information is requested for the displaying of the city map on the user’s mobile application. The operation of retrieving is quite simple, because it requires just the user position or an address, but the amount of data returned is high and needs some logic to be forwarded to the client. Thus, we decided to assign a high complexity value

* Address - Position Conversion Result

The Mapping Service is able to convert an address provided in String format to a pair of coordinates that will be used to update a single table on the database for each request, so the complexity of this EIF is low

* Money Saving Mode Data:

Money saving mode data is a collection of information that the user gets when activates that mode (ETA, Distance to the destination, Information about the path and so on). Although this data are computed by the mapping service, we need some logic operation in order to provide to the external service the information it requires (two pairs of coordinates, one for the user actual position and one for the Special Parking Area). Specifically, the choice of the second one is based on the final destination of the user, the availability of Special Parking Areas and the distribution of cars, so to jump to a conclusion we need several database queries, distance computation and comparisons. Taking into account these considerations, we decided to assign a medium (high) grade of complexity

* Distance Retrieval:

The distance between two pairs of coordinates is a simple data expressed with a double precision floating point value, computed by the Mapping Service. As inputs, that external component needs only the coordinates representing the start point and the end point. Since we only need to provide this kind of information (that we mainly store in the database) and we don’t have to apply any logic to the returning data (maybe just some formatting operation), we classify this operation with a low complexity

**Payment Service Provider:**

* Transaction Acknowledgement:

The only information we receive from the Payment Service Provider is an acknowledgment for the outcome of a user payment for a ride or a fee for an expired reservation. Since we store the payments in a single ILF, all we have to do with the data incoming from the Payment Service Provider is an update on a single field of a table on the database, we assume that the complexity of this operation can be evaluated as low

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| City Map Retrieval | High | 10 |
| Address/Position Convertion | Low | 5 |
| Money Saving Mode Data | High | 10 |
| Distance Retrieval | Low | 5 |
| Transaction Ack | Low | 5 |
| *Total* | | 35 |

### External Inputs (EIs)

In our PowerEnjoy, the back-end section interacts with different categories of client, providing a set of functionalities in response of several types of inputs.

In this section we will focus on the complexity of each EI (grouped by clients), explaining our considerations in every case of analysis.

EI Rating (Low = 3 FPs, Avg = 4 FPs, High = 6 FPs):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data Elements** | | |
| **File Type Referenced** | 1-4 | 5-15 | 15+ |
| 2- | Low | Low | Avg |
| 2 | Low | Avg | High |
| 2+ | Avg | High | High |

**End User:**

* Registration

The user registration involves the data processing by the Registration Manager, some, database checks, a record insert and the Notification Manager effort for sending the PIN. As we can notice, it is not a straightforward operation, so it can be considered with an average complexity.

* Login/Logout

The user Login & Logout operations are quite simple, as they affect just the Login Manager and include simple database operations (a check and a basic update). The overall complexity is low (3 FPs for each operation).

* Lost Password Retrieval

If a user loses its password, there are different steps in order to deliver a new one: database operation to delete and update fields, generation of a new one by the Account Manager and the Notification Manager to communicate it to the user. This complexity is evaluated as average.

* Car Reservation

This operation involves the Reservation Manager to perform a series of tasks, as it has to create a new Reservation entity based on the information provided by the user (driving license, car license plate) and instantiate a timer that determines the validity of a reservation for 60 minutes. It also has to update the availability of a car in the database. We assign an average level of complexity

* Car Identification

Identification of a car by a user is a complex operation. In fact, to achieve successfully this task, different components (and the respective managed entity on the database) have to be involved: the Identification Manager has to send a message to the Car Manager that forwards it to the right car, unlocking it physically. The Identification Manager has also to communicate to the Reservation Manager, in order to be sure that a car is actually available and to end a possible reservation made by the same user who performs the identification. Due to this considerations, we attribute a high level of complexity.

* Money Saving Mode Enabling

Money Saving Mode data is a collection of information that the user gets when activates that mode (ETA, Distance to the destination, Information about the path and so on). Although this data are computed by the mapping service, we need several logic operations in order to provide to the external service the information it requires (two pairs of coordinates, one for the user actual position and one for the Special Parking Area). Specifically, the choice of the second one is based on the final destination of the user, the availability of Special Parking Areas and the distribution of cars, so to suggest the right place to park we need several database queries, distance computation and comparisons. Taking into account these considerations, we decided to assign a high level of complexity.

**Car:**

* Generic Sensor Data Transmission

The data sensor transmission involves the Car Manager that receives and processes them to check if they respect some parameter (if the car is damaged or has some kind of issue. If not, it has to interact with the Assistance Manager for creating an automatic reporting that has to be stored in the database. The complexity is average.

* Ride Start Data Transmission

When a ride starts, the car transmits different data to the Car Manager (start position, battery level, number of passengers) which has to interact with the Ride Manager for the creation of a Ride entity. It is not a very complex operation, but it requires some effort due to checks and data collecting from different entities, so we assign an average level of complexity.

* Ride End Data Transmission

The end of a ride implies a high effort by various components the same way as for the ride start, but with the addiction of all the processes to determine the charge to be paid, including discount and overcharge calculation and the direct interaction with the Payment Manager. For this reason, we evaluated this complexity as high

**Assistance Team:**

* Telephone Assistance Requests Inserting

The assistance team’s task that include a direct interaction with the Central Server is the one regarding the insert in the system of the requests they receive via telephone. It is a simple insert or update operation, translated by the server in a database statement, so we rate it as a low complexity one.

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| Registration | Avg | 4 |
| Login/Logout | Low | 2x3 |
| Lost Password Retrivial | Avg | 4 |
| Car Reservation | Avg | 4 |
| Car Identification | High | 6 |
| Money Saving Mode Enabling | High | 6 |
| Generic Sensor Data Transmission | Avg | 4 |
| Ride Start Data Transmission | Avg | 4 |
| Ride End Data Transmission | High | 6 |
| Teleph. Assistance Req. Inserting | Low | 2x3 |
| *Total* | | 50 |

### External Inquiries (EQs)

Now we are going to analyse the External Inquiries (EQs) influencing our system, that are basically data retrieval request by an user, without a big amount of business logic being involved in the process. PowerEnjoy supports some of these kind of operations, everyone characterized by a low complexity level since here is not involved big amount of data and the effort for retrieval is quite low (except for the car retrieval, which needs the comparison between the user position and the car position by the Mapping Service):

* A user can request the available car in an area around its position or an inserted address
* A user can request the history of its past reservations
* A user can request the history of its past rides
* A user can request the history of its past payments (fees and ride charges)
* The Assistance Team can request assistance requests with several type of filters (by cars, users, zones, times etc)

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| Available Cars Request | Avg | 4 |
| Past Reservation Request | Low | 3 |
| Past Reservation Ride | Avg | 3 |
| Past Reservation Payments | Avg | 3 |
| Assistance Requests | High | 3 |
| *Total* | | 16 |

### External Outputs (EOs)

In the last part of the Function Point analysis we focus on the External Outputs, that are situation in which PowerEnjoy needs to communicate with the external environment, sending various types of information. This communications are:

* PIN notification to the user

A simple operation via E-Mail that involves just the Notification Manager and the Database Manager, to set up a right PIN creation. Complexity: low

* Fee notification to a user

It’s a simple E-Mail message sent to the user by the Notification Manager when a Reservation expires. Complexity: low

* Car Unlocking

It’s a simple outcome sent to the Car System by the Car Manager, to perform the unlocking. Complexity: low

* Fee and Ride Charge Payment Details

The Payment Manager sends this info to the Payment Service Provider. The system needs to collect some data in order to achieve this task (user’s billing details, amount to be paid), retrievable from few tables in the database. Complexity: average for both

* Assistance Request notification

The Assistance Manager has to collects sensors data passed by the Car Manager, and information about the car (license plate, position), attaching the type of problem (interpreting sensor’s data) and pass them to the Assistance Team System. Complexity: average

Overall FPs assignment:

|  |  |  |
| --- | --- | --- |
| **EIF** | **Complexity** | **FPs** |
| PIN Notification | Low | 4 |
| Fee Notification | Low | 4 |
| Car Unlocking | Low | 4 |
| Fee/Ride Charge Payment Details | Avg | 2x5 |
| Assistance Requests | Avg | 5 |
| *Total* | | 27 |

### Overall estimation

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

## COST AND EFFORT ESTIMATION: COCOMO II

### Scale Drivers

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

### Cost Drivers

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

### Schedule estimation

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

### Effort equation

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

### Schedule estimation

The assistance team handles every help request, as well as cars malfunctions and recharge issues. This information is provided to the assistance team by the central server (except for client help requests made directly with phone call), and the team has also the responsibility of storing and managing data retrieved by clients about the specific issue. In order to achieve that goal, assistance team’s terminals are equipped with a dedicated software for central server interaction.

The last component the central server interacts with is the Database system, with which communicate synchronously to store and retrieve the whole managed data.

# SCHEDULE

# RESOURCE ALLOCATION

This sequence diagram describes the procedure of registration into our application.

Once opened the app from the mobile phone, the user is asked to insert his account credentials. If the user doesn’t have an account yet, he has the possibility to enter in the registration section and insert his email address and driving license number. In the system both these information MUST be unique (a person can have only one account), so the system performs a verification in the database about the uniqueness of the data.

If data are unique, the system generates a unique code (PIN) and sends it back to the user. From that moment the PIN is the only way to the user to effectively identify cars.

In case the information is not unique, the user is notified that there is already an account using that email address or driving license number.

# RISK MANAGEMENT

Concerns future happenings

Involves hange in mind, opinion, actions, places etc..

Involves choice

Brainstorm:

* Other application on the market (already existing)
* Loss of external software (e.g. Google Maps, SMS gateway, Payment Methods)
* People leaving (difficult in this case, but heavy impact if happening)
* Change in available car zone

2 Tables:

1)

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Risk** | **Probability** | **Effect** |
| PR1 | A key member of the team quits or is ill in a critical point of the project development | Average | Serious |
| PR2 | Some parts of the project require more time than expected (underestimating schedules) | Average | Serious |
| PR3 | The external car software component doesn’t work as expected or is not ready in time | Low | Catastrophic |
| PR4 | Loss of entire or part of the source code | Low | Catastrophic |
| TR1 | External commercial components change their configurations/protocols/API | Low | Catastrophic |
| TR2 | A new paymenth method starts getting used widely | Average | Serious |
| BR1 | Change in available car parking zone | Average | Serious |
| BR2 | The project don’t satisfy the expectation of the city amministration | Average | Catastrophic |
| BR3 | Budget overestimation | High | Serious |
| BR4 | Budget underestimation | Average | Catastrophic |
| BR5 | Missing agreement with the mobile data vendor | Average | Catastrophic |

2)

|  |  |
| --- | --- |
| **Code** | **Mitigation strategy** |
| PR1 | Split duties and responsibilities among the highest number of people. In the worst case we can consider a first beta release with less functionalities than the defined ones during the requirement analysis phase |
| PR2 | Schedule an extra time slots before the major activities or releases |
| PR3 | A bunch of meetings with the external sw company to define deadlines |
| PR4 | Use of a backup system to store versions of data |
| TR1 | Try to build components that are independent among them (modularity of the system) with a particular focus on interfaces implementation |
| TR2 | Keep sistematically under control the new methods of payment on the market |
| BR1 | Exploit the time between the proposal and the confirmation of the measure to update our system |
| BR2 | Make the administration part of the development process (meetings and presentations on RASD and DD) |
| BR3 | Propose an alternative way of using the residual budget (application support, future releases, new functionalities) |
| BR4 | First release with less functionalities than the defined ones during the requirement analysis phase |
| BR5 | Exploit the economy of scale in the market of mobile data providers |

# EFFORT SPENT

## Giorgio Marzorati

29/11/2016 - 3h

04/12/2016 - 3h

05/12/2016 - 2h

08/12/2016 - 3h

09/12/2016 - 4h

10/12/2016 - 3h

11/12/2016 - 2h

## Aniel Rossi

29/11/2016 - 3h

04/12/2016 - 3h

05/12/2016 - 2h

08/12/2016 - 3h

09/12/2016 - 5h

10/12/2016 - 3h

11/12/2016 - 1h

## Andrea Vaghi

29/11/2016 - 3h

01/12/2016 - 2h

02/12/2016 - 3h

04/12/2016 - 1h

05/12/2016 - 2h

06/12/2016 - 1.30h

07/12/2016 - 1h

08/12/2016 - 3h

09/12/2016 - 1h

10/12/2016 - 5h

11/12/2016 - 3h

# CHANGELOG

V1.0 - First release