Politecnico di Milano  
Industrial and Information Engineering  
Computer Science and Engineering

Software Engineering II Assignment

PowerEnJoy - car sharing

Academic Year: 2016/2017

Prof. Elisabetta Di Nitto

Alice Segato matr. 875045

Mark Edward Ferrer matr. 876650

Davide Bonacina matr. 876199

PowerEnJoy



Car Sharing App

# TABLE OF CONTENT

[TABLE OF CONTENT 3](#_Toc468701611)

[1 INTRODUCTION 4](#_Toc468701612)

[1.1 Purpose 4](#_Toc468701613)

[1.2 Scope 4](#_Toc468701614)

[1.3 Definitions, Acronyms, Abbreviations (UNDER CONSTRUCTION) 4](#_Toc468701615)

[1.4 Reference Documents 5](#_Toc468701616)

[1.5 Document Structure 5](#_Toc468701617)

[2 ARCHITECTURAL DESIGN 5](#_Toc468701618)

[2.1 Overview 5](#_Toc468701619)

[2.2 High level components and their interaction 5](#_Toc468701620)

[2.3 Component view 6](#_Toc468701621)

[2.4 Deploying view 6](#_Toc468701622)

[2.5 Runtime view 6](#_Toc468701623)

[2.6 Component interfaces 7](#_Toc468701624)

[2.7 Selected architectural styles and patterns 7](#_Toc468701625)

[2.7.1 Overall architecture 7](#_Toc468701626)

[2.7.2 Protocols 7](#_Toc468701627)

[2.7.3 Design patterns 7](#_Toc468701628)

[2.8 Other design decisions 7](#_Toc468701629)

[3 ALGORITHM DESIGN 7](#_Toc468701630)

[4 USER INTERFACE DESIGN 7](#_Toc468701631)

[5 REQUIREMENTS TRACEABILITY 7](#_Toc468701632)

[6 REFERENCES 7](#_Toc468701633)

[6.1 Used tools 7](#_Toc468701634)

[7 HOURS OF WORK 7](#_Toc468701635)

# INTRODUCTION

## 1.1 Purpose

This document has the purpose to give more information about the PowerEnJoy System than the Requirement Analysis and Specification Document (RASD).

This document addresses developers and has the objective to identify:

* The architectural design;
* The design choices that we have made;
* How the system components interface with each other;
* The behavior of the system at runtime.

## 1.2 Scope

The system has the purpose of allowing users and more in general citizens to rent cars easily via mobile app in order to increase people’s mobility and decrease city pollution.

To use the application and the service that comes with it, people has to register and join a community of car sharers; after registration, they can rent cars and drive wherever they want to go with the condition to bring back the car in a safe area.

Users can choose a car from the nearest parking suggested by the app based on their GPS position or on the given address, and they can pick it up with a limited time.

Anytime during the driving, users can set the car in pit stop mode to park the car in a sort of “reserved” state where the car is still linked to the user but it is stopped and parked outside of the safe areas.

At the end of the ride, users must return the car in a safe area and the system calculates the final amount with respect to certain situations that give user discounts or surcharges: to help users with low budget, there’s also a money saving option that calculates the nearest special parking area to the final destination of the user to get the maximum amount possible of discounts.

Therefore, the final objective of the system is to allow the company to manage the car requests faster and automatically to substitute the previous system, described in section 1.2 of the RASD.

## 1.3 Definitions, Acronyms, Abbreviations (UNDER CONSTRUCTION)

All the words defined in the RASD at section 1.5 are still valid and they will appear in this document too.

DD: Design Document.

RASD: Requirements analysis and Specification Document.

## 1.4 Reference Documents

* RASD;
* Assignment AA 2016-2017.pdf;
* Sample Design Deliverable on Nov 2.pdf.

## 1.5 Document Structure

* **Introduction:** this section opens the document and shows the main purpose of the system-to-develop, the structure of the entire document and deepens some aspects introduced in the RASD.
* **Architecture Design:**
  + Overview: this section illustrates the physical deployment of our system;
  + High-level components and their interactions: shows how the different components of the system interface with each other and with the third-party system;
  + Component view: deepens the view of the components and gives more details;
  + Runtime view: this section explains with sequence diagrams, how the system should work during different tasks;
  + Component Interfaces: presents the communication of the different components;
  + Selected architectural designs and patterns: this section describes all the design patterns that we used to model the system and how they work.
  + Other design decisions: the title is self-explaining.
* **Algorithms Design:** this section includes some algorithms that manages particular tasks of the system. The algorithms are written in pseudo-code in order to clarify the behind the scenes of system with the maximum readability.
* **User Interface Design:** it should include design mockups to describe the possible result of the mobile application.
* **Requirements Traceability:** shows where the goals defined previously in the RASD take shape in design elements.

# ARCHITECTURAL DESIGN

## 2.1 Overview

[immagine overview]

As we can see our PowerEnJoy system has a 3-tier architecture:



**The presentation tier,** or user services layer, gives a user access to the application. This layer shows data to the user and optionally permits data input. The two main types of user interface for this layer are the traditional application and the Web-based application (used by the admin of third party system).

**The application layer** runs bussiness logic for both client sides (user and admin) to ensure that effective communication with instances of the client over the network is possible.

**The data layer**: This is the actual DBMS access layer. It can be accessed through the application layer. This layer consists of data access components to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC drivers on each client.

At this point we would like to point out some important observations:

* our server is going to be used by many different clients that we do not have control over.
* we want also to be able to update the server regularly without needing to update the client software.

So we need to minimize the coupling between client and server components in this application. In order to cover this requirements we decided to use RESTful API. RESTful web services, as the name suggests, are resources on the web that can be used to get specific information. These services basically portray the working of the REST API. The client requests a resource from the server and the server sends back the response.

In this way on the Client there will be not a static GUI but a dynamic one that is generated on client side.

## 2.2 High level components and their interaction

[immagine del component interactions]

Our system has four components that are the central system application, the client, the database and the car system. These communicate with the fifth component that is the third party system through the central system (they share the same database located in the Database server in order to give to both the same information about cars).

Users can send any type of message from the mobile application e.g. reservation message, support message and some others: central system handles all the incoming messages from all the clients and respond to everyone with a certain response. In order to allow users to do different actions simultaneously, the system is asynchronous so the communication has a certain delay due to the elaboration of multiple requests. The communication between these two components is bilateral so it allows system to send back messages to users, too.

The Car component communicates only with the central system and the communication is only in one direction: the car component acts like a passive element and it has only the task to send all the information retrieved with sensors (GPS included) and buttons to the system.

The third party system, as explained previously in the RASD, fulfills all the tasks linked to car sharing “bureaucracy” such as fine delivery to users, car maintenance, user support etc. It communicates only with the database that is shared between the two systems: in this way, in order to retrieve all the information about users, it queries the database directly without asking anything to the main system. We decided to use this type of communication to facilitate the installation of the new system and to maximize the compatibility with the already existing system.

Last but not least, the database component contains all information that is useful for both system such as car position at switch off, car data for maintenance, last user at switch of and so on (particularly, these attributes are useful for the system to show the cars in recovery state, the cars that have to be maintained and fine delivery to the users that got them).

## 2.3 Component view

[immagine component view]

* User.Authentication: handles the registration and the login of the user in the system;
* User.Positioning: handles the detection of user’s position;
* User.Reservation: handles all the actions linked to the reservation client side;
* User.Payment: handles the payment method that user inserts;
* System.UserHandler: handles multiple users and their authentication. Also allow them to see their personal information;
* System.ReservationHandler: handles the reservation server side. Checks if the user can rent a car or if a particular car can be rented;
* System.DataManagement: handles the communication between the MySQL server and the main system;
* Database: contains the MySQL database;
* Car.GPS: handles the detection of car position;
* Car.Sensor: handles all detection made by car sensors and sends them, together with the position, to the central system on car shut down.

## 2.4 Deploying view

[immagine del deployment ed eventuale descrizione]

## 2.5 Runtime view

[immagini dei sequence e spiegazione del funzionamento]

## 2.6 Component interfaces

[immagine component interfaces]

## 2.7 Selected architectural styles and patterns

[immagine class diagram]

### 2.7.1 Overall architecture

### 2.7.2 Protocols

### 2.7.3 Design patterns

## 2.8 Other design decisions

# 3 ALGORITHM DESIGN

# 4 USER INTERFACE DESIGN

# 5 REQUIREMENTS TRACEABILITY

# 6 REFERENCES

## 6.1 Used tools

# 7 HOURS OF WORK