Power EnJoy
Design Document

Niccolo' Raspa, Matteo Marinelli

December 4, 2016

Software Engineering 2 Course Project

Contents

1 Introduction

1.1 Purpose

This is the Design Document for the Power Enjoy Service. It's aim is to provide a functional description of the main architectural components, their interfaces and their interactions, together with the algorithms to implement and the User Interface Design. Using UML standards, this document will show the structure of the system and the relationships between the modules. This document is written for project managers, developers, testers and Quality Assurance. It can be used for a structural overview to help maintenance and further development.

1.2 Scope

PowerEnjoy is a digital management system for a car-sharing service that exclusively employs electric cars. It allows registered clients (Power Users) to use a vehicle paying only on the basis of the actual use during each individual rental. For a more detail description of the domain and the requirements please refer to the Requirement and Specification Document.

The software system is divided into four layers, which will be presented in the document. The architecture has to be easily extensible and maintainable in order to provide new functionalities. Every component must be conveniently thin and must encapsulate a single functionality (high cohesion). The dependency between components has to be unidirectional and coupling must be avoided in order to increase the reusability of the modules.

Futhermore, to increase cohesion and decoupling as much individual components must not include too many unrelated functionalities and reduce interdependencies.

1.3 Definitions, Acronyms, Abbreviation

RASD: Requirements Analysis and Specification Document.

DD: Design Document.

DBMS: Relational Data Base Management System.

DB: Database layer,

UI: User Interface.

Backend: Term used to identify the Application server.

Frontend: The components which use the application server services (web frontend and the mobile applications).

SOA: Service Oriented Architecture.

JDBC: Java DataBase Connectivity.

JPA: Java Persistence API.

EJB: Enterprise JavaBean.

ACID: Atomicity, Consistency, Integrity and Durability.

1.4 Reference Documents

This document refers to the following documents:

- Project rules of the Software Engineering 2 project
- Requirement Analysis and Specification Document (from the previous delivery)

1.5 Document Structure

This document is structured in five parts:

Chapter 1: Introduction. This section provides general information about the DD document and the system to be developed.

Chapter 2: Architectural Design. This section shows the main components of the systems with their subcomponents and their relationships, along with their static and dynamic design. This section will also focus on design choices, styles, patterns and paradigms.

Chapter 3: Algorithm Design. This section will present and discuss the main algorithms for the core functions of the system, independently from their concrete implementation.

Chapter 4: User Interface Design. This section shows how the user interface will look like and behave, by means of concept graphics and UX modeling.

Chapter 5: Requirements Traceability This section shows how the requirements in the RASD are satisfied by the design choices, and which compontents will implement them.

2 Architectural Design

2.1 Overview

This chapter provides a comprehensive view over the system components, both at a physical and at a logical level. This description will follow a top-down approach, starting with the description of the high-level components and their relations and interactions. We will then reason and describe the single components and the functionalities they must implement. We will especially focus on the components that implement the core logic of our application and using sequence diagrams we will describe the runtime behaviour of the system.

We will also include deployment diagrams to show the physical implementation of the system.

2.2 High Level Components

Before describing the actual system architecture we introduce the high level components of our application. Following the requirements and the specification listed on the RASD, we identify what components are needed in order to implement them and only after we have detected them, we will focus on the architecture and explain our architectural decision and the technlogies chosen. It's important to follow this process to generalize our design as much as possibile and abstract from implementation details, in this way we are able to describe only the essence of our system.

The main high level components of the system are the following:

Mobile Application: Power Enjoy is a car sharing service therefore it must be implemented with mobility in mind. Since the majority of the mobile devices have a GPS module and we need to have access to the user position for our application, it makes sense to require that the main user has our mobile application installed.

Application Server: This component contains all the logic for the system application. It will implement all the required functionalities and communicate both with the mobile application and the external services.

Web Server: This component does not contain any application logic, it's used to provide a web interface interface to the user. It helps to separate presentation from logic.

Web Browser: Using a web browser the user is able to communicate with the Web Server to obtain the required web pages.

Database: This components is responsible for data storage and retrieval which is crucial for our application. It will not implement any logic but it stores all the information needed for the correct functioning of our service. It must guarantee ACID properties and be accessible from the Application Server.

To give a complete overview of the system, we list also the external services which are not part of our system but with which the system depends to implement some functionalities (please refer to the RASD for a more detailed description).

High level components which are not part of our system:

Assistance Service: Power Enjoy is in charge of the management of the car-sharing system. All the secondary functionalities are handled by an assistance service. This components provides an API to handle all the assistance request.

Payment Service: Every Power Enjoy user, in order to use our service, is required to have an account registered to a third party payment service, with a valid payment method. This components provides an API to handle all the payments functionalities.

Car On Board System: Every Power Enjoy vehicle comes with a pre-installed on board system which registers and notifies all the car activity. This components provides an API to monitor and remotly control every vehicle.

In the figure below we rapresent all the components listed above in a layered fashion, highlighting the relations among the different parts:

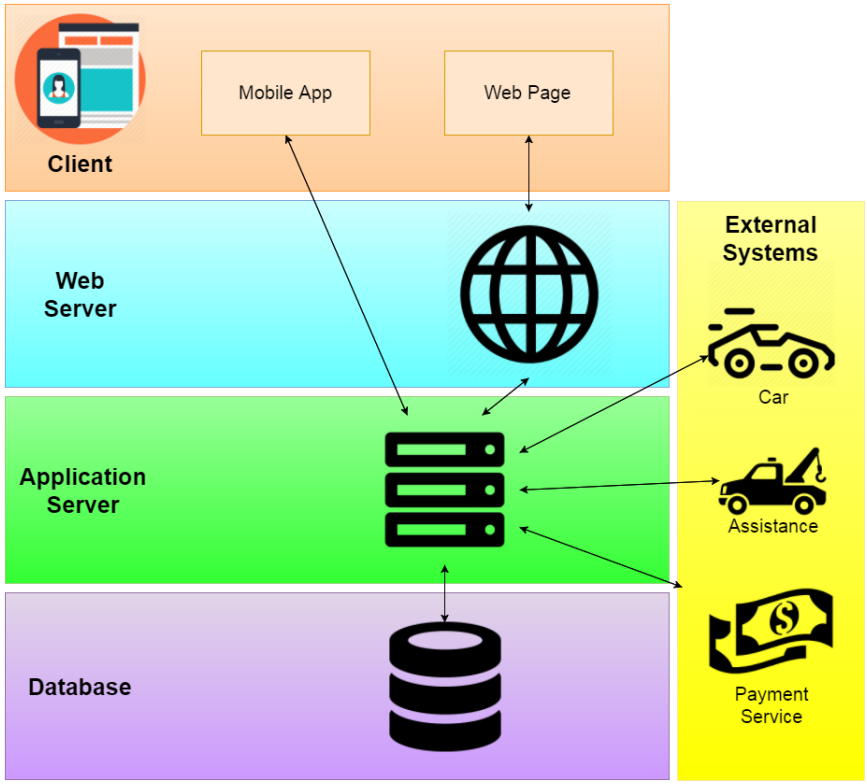

Figure 1: High Level Components

From this diagrams it's important to point out:

- Separate Application Server and the Web Server improves scalability, since there may be many web servers talking to a single application server. Further implementations may include the use of caching at the web server level. distinct physical tiers that can individually be optimized to perform their respective task.
- This layered rapresentation already suggest a four tier architecture,
- The Application Server is the bottleneck of our system. Every other components is in relation with it, therefore it's performance is stricly related to the performance of our system.

2.3 Component View

In this section the individual components will be discussed in terms of the needed high level sub-parts and their functioning, as well as how those sub- elements interface with one another within the overlaying component and which of them is in charge of interfacing with other components.

2.4 Deployment View

lorem ipsum lorem ipsum

2.5 Runtime View

Sequence diagrams to descrive the way componenets interact to accomplish specific tasks typically related to your use cases

2.6 Component Interfaces

2.7 Selected Architectural Styles and Pattern

Explain styles/patterns used and why/how

2.8 Other Design Decision

3 Algorithm Design

Focus of the most relevant algorithmic part

4 UI Design

Overview of how the user interace of your system will look like

5 Requirements Traceability

Explain how requirements defined in the RASD map to the design elements that you have definited in this document

6 Effort Spent