Politecnico di Milano, A.A. 2016/2017

Software Engineering 2: PowerEnJoy
Requirements **A**nalysis and **S**pecification **D**ocument

Binosi Lorenzo 876022

November 7, 2016

Contents

1 Introduction

1.1 Purpose

This is the Requirement Analysis and Specification Document for the PowerEnJoy car-sharing service. Its purpose is to completely describe the system, its components, functional and non-functional requirements, constraints, and relationships with the external world, and to provide typical use cases and scenarios for all the actors involved. It is also a strong baseline for project planning and cost estimization, for software evaluation and change control.

This document is written for project managers, developers, testers and systems analysts. Users are not generally interested in detailed software requirements, but they can still find it useful. It may be used in a contractual requirement.

1.2 Scope

PowerEnJoy is a digital management system for a car-sharing service that exclusively employs electric vehicles. This car rental system provides an alternative solution to public transport, thus being not only eco-friendly, but also simple and reliable.

The service is available to whomever has a valid driving license. Anybody who wants to register must provide a copy of it, along with his credentials and payment informations. Once registered, they will be allowed to search a vehicle in a specific area, and reserve one for up to an hour before picking it up. The system will cancel the reservation in case of them failing to arrive to the vehicle in time, will make that specific vehicle available again and apply a fine to the driver. Upon arrival to the vehicle, the driver will be able to get inside and drive it, at the cost of a per-minute fee of which he will be notified by the system through a screen on the vehicle itself; this until the vehicle will be left parked in a safe area, where the system will close the doors and make the vehicle available again.

A support team is provided by the service in case of incidents, vehicle failures and other cases of need.

1.3 Actors

- Guest: anyone not recognized by the system who can sign up or log in to it. He/She may look for informations about the service.
- Driver: a user subscribed to the system.

- Operator: a user that is working for, and subscribed by, the system.

1.4 Goals

The goals of the PowerEnJoy service are the following:

- [G1] Allow a driver to rent a vehicle.
  - [G1.1] Allow a driver to search and reserve a vehicle.
  - [G1.2] Allow a driver to use a reserved vehicle.
  - [G1.3] Allow a driver to park a vehicle in a safe area, suggested by the system.
- [G2] Notify the driver about every fee, discount, or sanction/fine that will be applied to him while using the service
- [G3] Allow a guest to signup to the system.
- [G4] Motivate a driver into car pooling.
- [G5] Motivate a driver to a better behavior.
- [G6] Provide an efficient and fast assistance in case of problems to vehicles.

1.5 Definitions, acronyms, and abbreviations

RASD: Requirements Analysis and Specification Document.

System: the whole software system to be developed, comprehensive of all its parts.

Car pooling: the sharing of car journeys so that more than one person travels in a car.

User: anyone recognized by the system. In this case drivers or operators.

RDBMS:

API:

JVM:

1.6 References

This document refers to the project rules of the Software Engineering 2 project [1] and to the RASD assignment [2].

This document follows the IEEE Standard 830-1998 [3] for the format of Software Requirements specifications.

1.7 Overview

This document is structured in three parts:

section 1: Introduction. It provides an overall description of the system scope and purpose, together with some information on this document.

section 2: Overall description. Provides a broad perspective over the principal system features, constraints, and assumptions about the users and the environment.

section 3: Specific requirements. Goes into detail about functional and nonfunctional requirements. This chapter is arranged by feature.

2 Overall Description

2.1 Product Perspective

PowerEnJoy system is a simple client-server architecture based on a back-end server application and diffent front-end client applications, supported by different operating systems.

2.1.1 User Interfaces

Guests and users can interact with the service via the web application or the mobile application. Drivers can find other service functionalities in the car application. It is necessary to provide a common and uniform look and feel among the different hardware architectures.

All the interfaces shall be intuitive and user friendly. They should not require the reading of detailed documentation to be used.

2.1.2 Hardware Interfaces

The main hardware interface is a dedicated embedded system, installed in any vehicles, provided of several plugins and a touchscreen display for interact with the car application. Thanks to this embedded architecture the system will be also notified about the status of the vehicle and its location, even if the main battery is completely discharged.

2.1.3 Software Interfaces

Mobile application and web application are supposed to be friendly with any device, in particular the first one must be developed for iOS and Android, and the second one will work on any operating system that support a web browser.

On the embedded system of any vehicles is installed a JVM that runs a Java application that provides informations to the driver and to the system.

The back-end server stores its data in a RDBMS and can run on every platform that supports the JVM. It also provides different APIs for different functions that a user, or a guest, can do through client applications.

2.2 Product Functions

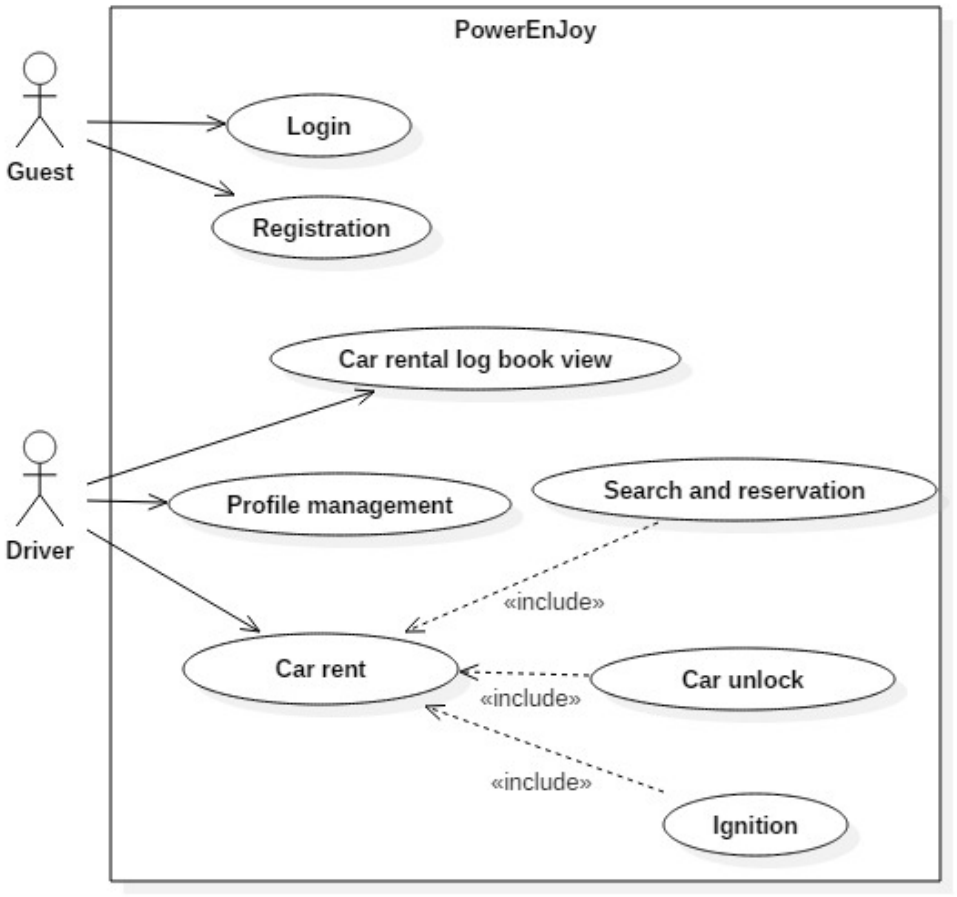

Figure 2.1: The comprehensive use-case diagram of all the functionalities implemented by the system.

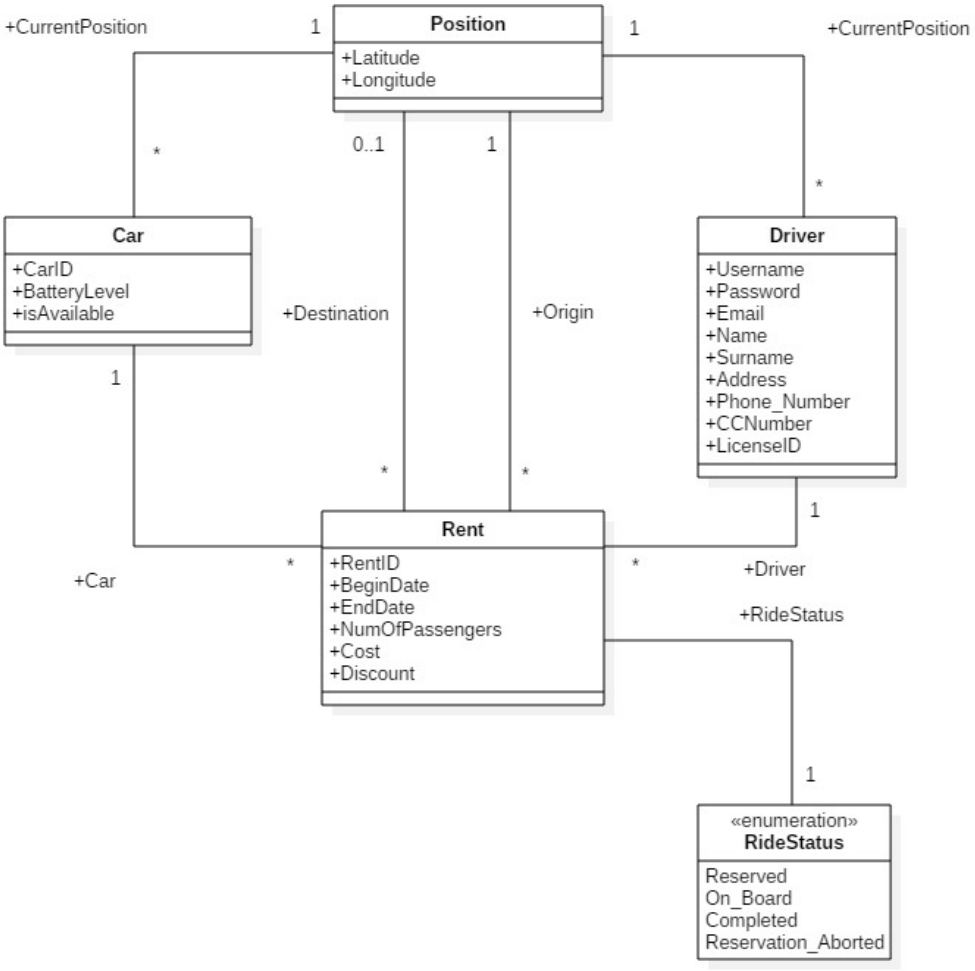

Figure 2.2: The comprehensive class diagram of the system.

The following lists reassume what anyone can do interacting with the system.

- Guests can:
  - create an account (signup to the system)
  - login
- Drivers can:
  - edit profile informations
  - delete their account

  - rent a vehicle
  - check their status
  - access to the vehicle rental log book

- Operators can:
  - solve vehicle problems

2.3 User Characteristics

The two kinds of users are drivers and operators. Only drivers can registered to the system independently, while operators must obtain a contract of employment with the PowerEnJoy service to access as operators. The system provides special accounts for them and they would be able to access to their job functionalities through the same, web and mobile applications, of drivers. They will be recognized by the system and a special view will be provided to them.

It's granted that both kinds of users have access to Internet.

2.4 Constraints

2.4.1 Regulatory policies

Any driver must follow current traffic and regulamentation laws of the area where the system is operating its car-sharing service. Driving licenses from other countries must be compatible with the laws of the country where the service is operating. Aggiungere regolamentazioni di lavoro

2.4.2 Service policies

- The system must ask the user for the permission to acquire, store and process personal data and web cookies.
- A driving license can't be associated to multiple driver accounts.
- Drivers with status of pending payments can't delete their accounts and can't access to rental functions.
- Drivers with invalid/expired driving license can't access to rental functions
- Drivers can rent a single vehicle per time.

2.4.3 Hardware limitations

The service requires an Internet connection fast enought to guarantee a fast response from the server and hardware architectures that can run properly the client side applications (web and mobile apps).

2.4.4 Reliability requirements

The system must have a minimum availability of 98%.

2.4.5 Parallel operations

The system must support parallel operations from different drivers that may require access to the database.

2.5 Assumptions and Dependencies

It assumes that:

- All users have access to a stable Internet connection.
- All drivers have a valid credit card.
- The GPS on the vehicle is working correctly.
- The driver specifies the correct location, if its GPS is not available.
- Vehicles model is the same, so any cars will have the same number of seats.
- Vehicle's sensors can detect correctly the number of passengers.
- The System applies a correct discount at the end of the renting and than charges the drivers directly via credit card.
- The System provides to make a vehicle available again if will pass more than one hour from the reservation time.
- The number of vehicles is sufficient to satisfy the demand in each area.
- Any fine taken by a driver will be paid by the system that will charge him/her for the same fine plus an additional fine provided by the system.
- All the areas where the system is operating are covered by a reliable 3G/4G connection.

- Operators are arranged in convenient locations that allow them to be efficient and fast when recovering and charging vehicles.
- The system is operating in an area composed by a city or agglomerated cities, of the same country, closed each other.
- The application on the vehicle will automatically notify the system about incidents, vehicle failures or low charges. If the application will stop to dialogate with the server, the system will considered it like a notification of vehicle failure in the last position assumed by the vehicle.

2.6 Future Extensions

The system will be implemented foreseeing the possibility of further extensions, for example:

1. If a vehicle is left at special parking areas where they can be recharged and the driver take care of pluggin the vehicle into the power grid, the system applies a discount of 30% on the last ride.
2. If a vehicle is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate to compensate for the cost required to re-charge the vehicle on-site.
3. If the driver enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the vehicle to get a discount. This station is determinate to ensure a uniform distribution of vehicles in the city and depends both on the destination of the driver and on the availability of power plugs at the selected station.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User interfaces

The user interfaces must satisfy the following UI constraints:

- Web application
    1. The web pages must adhere to the W3C standards. In particular, the software shall conform to the HTML 5 [4], CSS [5] standards.
- Mobile application
    1. The iOS version must adhere to the iOS Human Interface Guidelines [6].
    2. The Android version must follow Android design guidelines [7].
- Common to web and mobile applications:
    1. The client applications must have an UI that is accessible to disabled people.
    2. The interface must offer the possibility to choose the language used at all times.
    3. The first screen must ask the user to login in order to begin operations.
    4. UI controls and views must be suitable for the input interface and the screen size.
- Car application
    1. Radio and navigation applications must be implemented.
    2. A Label on the screen will always notify the driver about the current cost of the ride.
- Server back-end
    1. The server back-end must be configurable by means of a configuration text file.

3.1.2 Hardware interfaces

The embedded system of any vehicles must be provided of

- a 7" touchscreen display
- a GPS device
- sensors that check if all the parts of the car are working correctly
- sensors that check how many passengers are on the car
- a 4G router for a stable Internet connection
- a secondary battery used only if the main battery is discharged. The vehicle can't use this battery

3.1.3 Software interfaces

The required software products used by the back-end are:

- MySQL 5.7[1]
- Java SE 8[2]

The required software product used by the car application is:

- Java Embedded[3]

The required operating systems for the mobile application atr:

- iOS 8 or more recent
- Android 6.0 or more recent

For a detailed specification of the programmatic interfaces, see **??**.

3.1.4 Communications interfaces

The clients communicate with the server via HTTPS requests (port 443).

[1] http://dev.mysql.com
[2] http://www.oracle.com/technetwork/java/javase/overview/index.html
[3] http://www.oracle.com/technetwork/java/embedded/overview/index.html

3.2 System Features

3.2.1 Driver registration

3.2.1.1 Purpose Any guest can subscribe through web or mobile applications.

In both cases the guest has to fill a registration form and must agree to the personal data policy according to his/her country privacy laws, otherwise the registration request shall be aborted.

As soon as the guest has submitted all the data, the system verifies the consistency of the information and a confirmation mail with a password is sent to the email address indicated in the registration form. The guest must confirm his/her email address for end the registration.

Once registered a guest can be recognized as a driver through the login phase.

3.2.1.2 Scenario 1 Bob, a normal citizen without a car, has just discovered the existence of the PowerEnJoy Service and he wants to use it.

He opens the homepage of PowerEnJoy website and procedes to the registration clicking on the button "sign up".

He gives all the information required and authorises the personal data treatment.

The system verifies all the informations that bob submitted to the form and sends a confirmation mail to him.

Bob checks his mailbox, opens the mail and clicks on "Confirm email". He is also notified about his current password.

The system informs Bob that he is successfully registered.

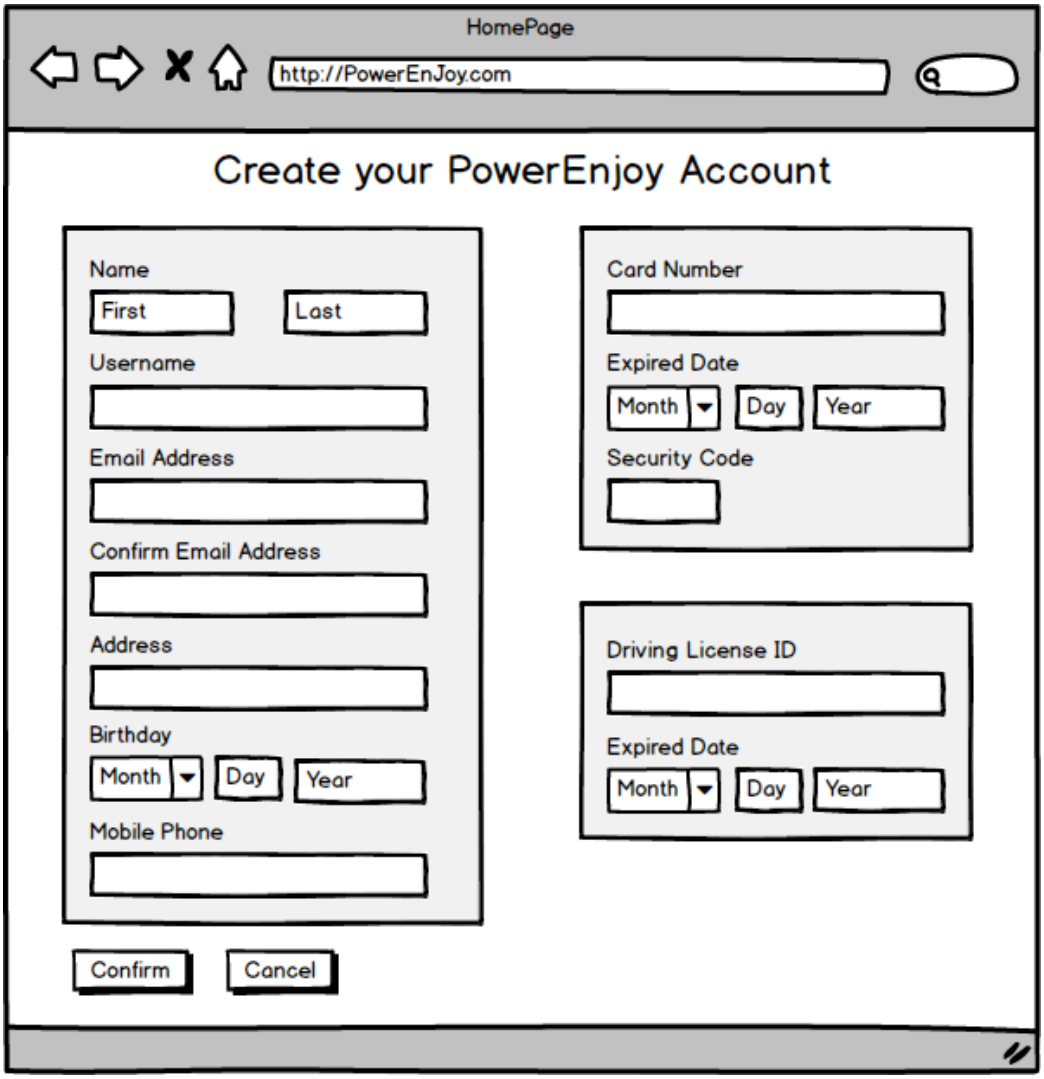

Figure 3.1: Concept of the registration webpage.

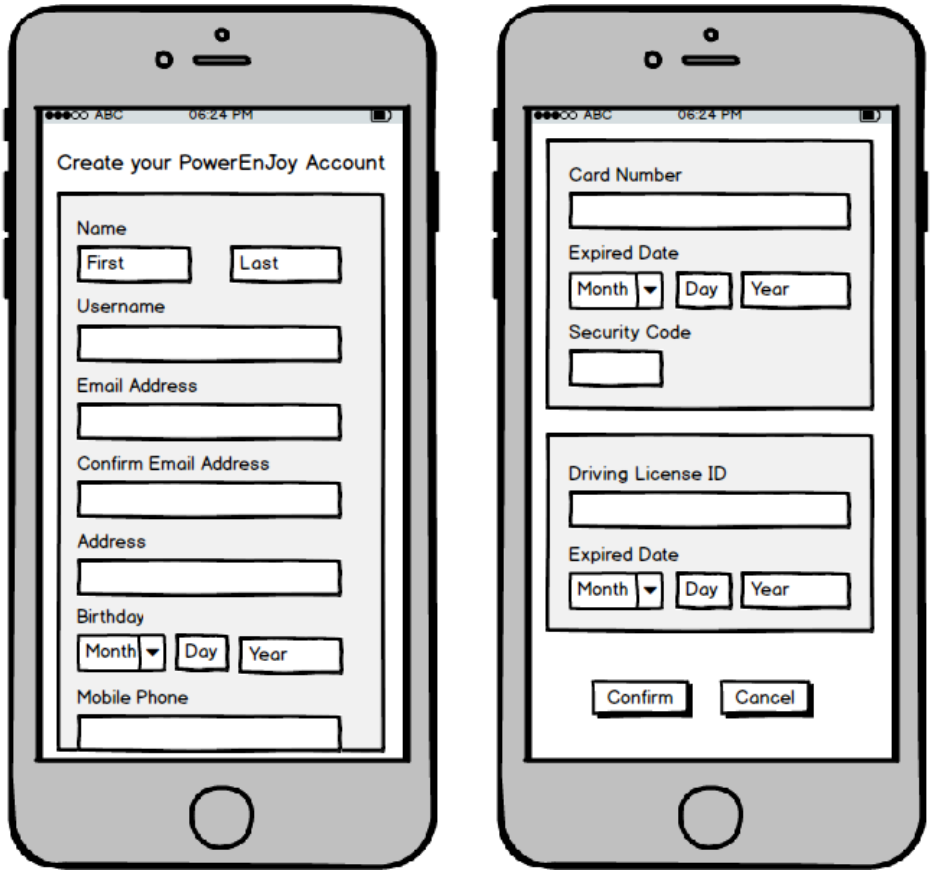

Figure 3.2: Concept of the mobile registration.

3.2.1.3 Use case description

3.2.1.4 Associated functional requirements

3.3 Performance Requirements

3.4 Software System Attributes

3.5 Alloy

4 Appendix

4.1 Used tools

To create the RASD we used the following tools:

- **MikTex**
  Distribution of the typesetting system LaTeX
  http://miktex.org/
- **TexStudio**
  OpenSource cross-platform LaTeX editor we used to write the RASD.
  http://texstudio.sourceforge.net/
- **StarUML**
  UML modelling tool we used to build the graphs
  http://staruml.io/
- **Balsamiq**
  The mockup builder we used to design the mockups.
  https://balsamiq.com/
- **Alloy analyzer 4**
  used to build strong and substantial models
  http://alloy.mit.edu/alloy/
- **GitHub desktop**
  Desktop application of the web-based Git repository hosting service. Used to collaborate in the team and to have a track of the changes.
  https://desktop.github.com/

4.2 Hours of work

This is the time spent redacting the RASD

- Lorenzo Binosi - 3 hours

References

[1] Software Engineering 2 Project, AA 2016/2017 - *Project goal, schedule and rules*

[2] Software Engineering 2 Project, AA 2016/2017 - *Assignments 1*

[3] IEEE Standard 830-1998: *IEEE Recommended Practice for Software Requirements Specifications*

[4] W3C, *HTML5 - W3C Recommendation 28 October 2014*, http://www.w3.org/TR/html5/.

[5] W3C, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, http://www.w3.org/TR/CSS21/

[6] Apple, *iOS Human Interface Guidelines*, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

[7] Google, *Android Developers - Design*, https://developer.android.com/design/index.html