Figure 1: Politecnico di Milano

Integration Testing Document
Version 1.0

Emanuele Ricciardelli (mat. 875221)
Giorgio Tavecchia (mat. 874716)
Francesco Vetró (mat. 877593)

January 13, 2017

Contents

1 Introduction

1.1 Purpose

This document represents the Integration Testing Plan Document for PowerEnJoy service. The purpose of the testing is to verify and ensure that the system operation is correct and complies with all the requirements expressed in the previous documents (RASD and DD). All components of the system will have to comply with the constraints and functionalities for which they were designed, without showing unexpected behavior. The purpose of this document resides therefore in specifying the methods and the time needed to complete a full testing, both from the point of view of the individual components, both in their interactions. In order to provide a detailed description of the future testing, in the following paragraphs it will be defined:

- A list of the subsystems and their subcomponents of the PowerEnJoy system-to-be that will be involved in the testing activity;
- The criteria that must be met by the project status before integration testing of the outlined elements may begin;
- A description of the integration testing approach and the rationale behind the specific choose;
- The sequence of component and function integration;
- A description of individual steps and test description: for each step of integration, we will define a test design with related expected results for each test;
- A list of tools, test equipment and test data required.

1.2 Definitions and Abbreviations

- RASD: Requirement Analysis and Specification Document;
- DD: Design Document;

For all other definitions, please refer to previous documents listed in the next paragraph

1.3 Reference Documents

- Specification document: Assignments AA 2016-2017.pdf
- RASD Version 1.1
- DD Version 1.1
- Integration testing example document.pdf

2 Integration strategy

2.1 Entry criteria

With reference to the past documents and in particular to the design document, it is necessary to specify how the individual components that make up the system require to achieve a given percentage of completion with respect to the features for which they were designed. Especially, with respect to the High Level Component view of the Design Document:

- 100% for the DBMS component: in order to allow a correct testing not only of the database structure itself, but also of the various components of the system that base their operation on access to the database;
- 60% of the Client components: many of the functions covered by the client involve the GUI and have no priority over interaction with the system functionality;
- 90% of the Application Server;
- 90% of the Car Sensor Subsystem: in order to ensure a correct testing of the application server, given the close interaction.

These criteria must be met before the start of the Integration testing. Different percentages want to reflect the order of integration, placed its focus on certain components, and otherwise in the background some secondary aspects such as the graphical user interface on the client.

2.2 Elements to be integrated

In order to specify the elements to be integrated, we refer not only to High Level Component view contained in the DD, but also to the Component view that allows us to identify components at a lower level and the relationships between them . As specified in the Design Document, the components of our system have been designed in such a way as to ensure a high level of modularity, by separating the different functionality into distinct components, thus also facilitating testing. For this reason, right from the Component view it is possible to note how different components are strongly dependent on one another in order to achieve more complex functionality of the system-to-be. The recognition of these complex functionality allows us to split our system into a number of subsystems, addressed respectively to the access management and registration system (*Account Subsystem*), management of payments (*Payment subsystem*), reservation management and everything that concerns the sphere of rental (*Reservation subsystem*) and finally management of all the sensors and the exchange of information between cars and system and everything related to the maintenance (*Car / Sensors subsystem*). In particular, the functions relating to the *Reservation subsystem* will involve the integration of:

- Reservation Controller;
- Ride Controller (integrated with Google Maps API);
- Notification Helper (with related gateway);

- Communication Manager;
- Request Manager;
- Data Access Manager.

With respect to the *Account subsystem*, it is concentrated only in the *Login / Registration Controller* that later, through the *Communication Manager*, will interface with the other components of the PowerEnJoy system. In detail:

- Login/Registration Controller;
- Communication Manager;
- Request Manager;
- Notification Helper (with related gateway);
- Data Access Manager.

The same reasoning can be made for the *Payment subsystem* and the respective *Payment Controller*. In detail:

- Request Manager;
- Payment Controller;
- Communication Manager;
- Data Access Manager;
- Payment Gateway;

Finally, the *Car / Sensors subsystem* in order to perform all its functionalities will need to integrate:

- Cars Controller;
- Communication Manager;
- Collector Data (with related gateway);
- Maintenance Gateway;
- Request Manager;
- Data Access Manager;

It is important to underline that the components *Notification Helper Request Manager* and *Communication Manager* are significant for all subsystems and therefore their integration. This means that if on the one hand their presence ensures greater flexibility in the management of functions, on the other hand emphasizes the importance of their proper functioning in order to ensure the fulfillment of all the functionality of the overall system. Given the above definitions of subsystems, our integration testing will proceed initially considering a single subsystem at a time, following finally with the integration of the same until the establishment of the system in its entirety.

2.3 Integration testing strategy

The approach suggested in the process of integration testing is bottom-up. This choice is based on considerations relating to the features covered by our system: in particular, as has been documented in the DD, it has been chosen to decompose the system into a series of components, each with features related to a particular field. The functionalities that the system will perform are derived from joint work of these components. From this observation it can be seen that the choice of a bottom-up reflects the needs of the testing related to our system. In this way we can favour the ability to identify problems at an early stage of development of the components and, in case, to react quickly. Later, after having tested the elementary interactions between components and subsystems, we can proceed with the integration to a higher level. It will therefore be necessary to develop a set of drivers to drive the testing, but given the simultaneous development of the overall system, this will prove to be quite natural. Once the system will be completely formed, we shall provide the implementation of a series of System Testing in order to verify not only the achievement of the functional requirements, as a whole, but also of the non-functional aspects such as the loading capacity as a function of the expected values.