POLITECNICO
MILANO 1863

15 January 2017

PowerEnJoy

Integration Test Plan Document

Blanco	Federica	875487
Casasopra	Fabiola	864412

Software Engineering 2 Project
2016/2017

Contents

List of Figures

List of Tables

1 Introduction

The Integration Test Plan Document aims at describing the planning in order to accomplish the integration test for our application PowerEnJoy. This document is useful for the development team, which is responsible for the creation of the integration test scripts in accordance to what is described in the next sections. Moreover, a developer will be chosen and he will be responsible for execution of the test scripts and certifying that the integration testing is complete. Furthermore, integration testing includes interactions between all layers of an application, including interfaces to other applications, as a complete end-to-end test of the functionality.

1.1 Revision History

`Version 1.0`, on 15 January 2017.

1.2 Purpose and Scope

The aim of the project PowerEnJoy is to provide a car-sharing service that involves *only* electric cars. In this documents, what to test, in which sequence, which tools are needed for testing and which stubs, drivers or oracles need to be developed is explained. If you wish to have more details about the scope of our project, you may refer to the *Section 1* of the Requirements Analysis and Specifications Document.

1.3 List of Definitions and Abbreviations

Here there is the acronims and abbreviations list:

DD Design Document

GPS Global Positioning System

ITPD Integration Test Plan Document

IT Integration Test

RASD Requirements Analysis and Specifications Document

TP Test Procedure

1.4 List of Reference Documents

- Specification document: Assignments AA 2016-2017.pdf

- IEEE Std 1016tm-2009 Standard for Information Technology - System Design - Software Design Descriptions.
- Requirements Analysis and Specifications Document: RASD.pdf (`https://github.com/fabiola-casasopra/sw-eng-2-project/tree/master/RASD/RASD.pdf`)
- Design Document: DD.pdf (`https://github.com/fabiola-casasopra/sw-eng-2-project/blob/master/DD/DD.pdf`)

1.5 Document overview

Here we show the structure of the document, with a brief overview of each section.

Section 1 There is an introduction with this document's purpose and other general information about it.

Section 2 There is the definition of all the items to be tested and the explanation of the integration testing approach.

Section 3 Here, for each step of the integration process above, there is a description of the type of tests that will be used to verify that the elements integrated in this step perform as expected. Moreover, ther is a general description of the expected results of the test set.

Section 4 Here, we are going to identify all tools and test equipment needed to accomplish the integration and there will be an explanation on why and how we are going to use the specific tool.

Section 5 Here, we are going to identify any program stubs or special test data required for each integration step, referring to the testing strategy and test design described in the previous section.

Section 6 Here there are given additional information that may be useful to the reader, such as the tools used and the time spent to redact this document.

2 Integration Strategy

2.1 Entry Criteria

In this part of the document, we are going to specify the criteria that must be met before integration testing of specific elements may begin:

- The Requirements Analysis and Specifications Document and the Design Document must be already completed, in order to know the interaction of the various components and their expected behaviour;
- Each component of our software must have successfully passed the Unit Testing;
- So, the correct version of our application is moved into the integration testing environment;
- All the code of our project must be already written and so the major functionality must be present;
- Our project should satisfy the memory requirements specified in the RASD;
- The database should be ready and its tables should already be populated with the initial data.

2.2 Elements to be Integrated

As we have shown in the Design Document related to our project PowerEnJoy, the system relies on many high-level components, each one implementing a specific set of functionalities, that interacts between them. Since we have decide to follow a modular approach, each components is the result of the combination of various subcomponents. However, since we haven't fully defined all low level component needed for our system, we think it is better to focus our integration testing only on the Business Logic and its components (for further information, see *Section 2* of the Design Document). By doing this choice, we have to consider that, in the following evolution of our project, the needed subcomponents will be created and further Integration Test must be carried out.

So, for what we said above, the elements to be integrated are the following:

- Web Component and Business Logic Component, testing the direct connections between Managed Beans and their corresponding Managers;
- the subcomponents of the Business Logic Component, integrating them, each one with the needed others.

2.3 Integration Testing Strategy

As we explained above, in this stage of the development we haven't fully defined the hierarchy of all subcomponents and subsystem. For this reason, we will have an Integration Test strategy for a single abstract layer and we have to keep in mind that other lower level subcomponents will be implemented. Although it is not possible to define the final integration test strategy, we think that, as far as we know at this stage, the better strategy we can apply is the top-down approach. Moreover, the chose of this strategy allow us to test the new subcomponent folowing the downward development.

2.4 Sequence of Component/Function Integration

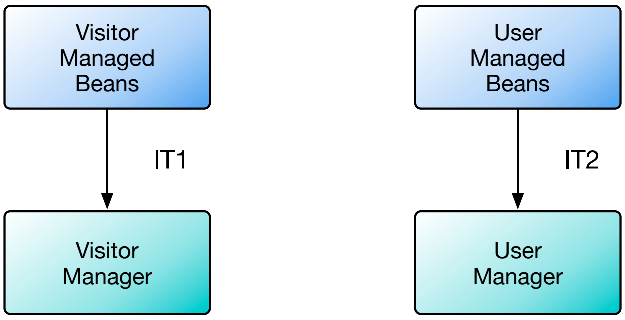

Figure 1: Managed Beans and corresponding Managers connections

ID	Components	IT	TP
IT1	Visitor Managed Beans → Visitor Manager	3.1.1	3.2.1
IT2	User Managed Beans → User Manager	3.1.2	3.2.1

Table 1: Managed Beans and corresponding Managers connections

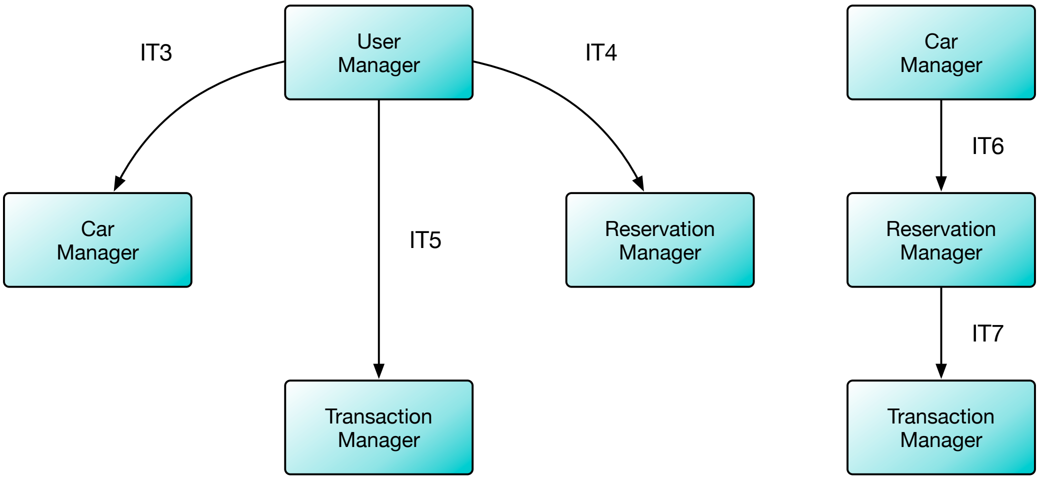

Figure 2: Business-tier subcomponents connections

ID	**Components**	**IT**	**TP**
IT3	User Manager → Car Manager	3.1.3	3.2.2
IT4	User Manager → Reservation Manager	3.1.4	3.2.2
IT5	User Manager → Transaction Manager	3.1.5	3.2.2
IT6	Car Manager → Reservation Manager	3.1.6	3.2.2
IT7	Reservation Manager → Transaction Manager	3.1.7	3.2.2

Table 2: Business-tier subcomponents connections

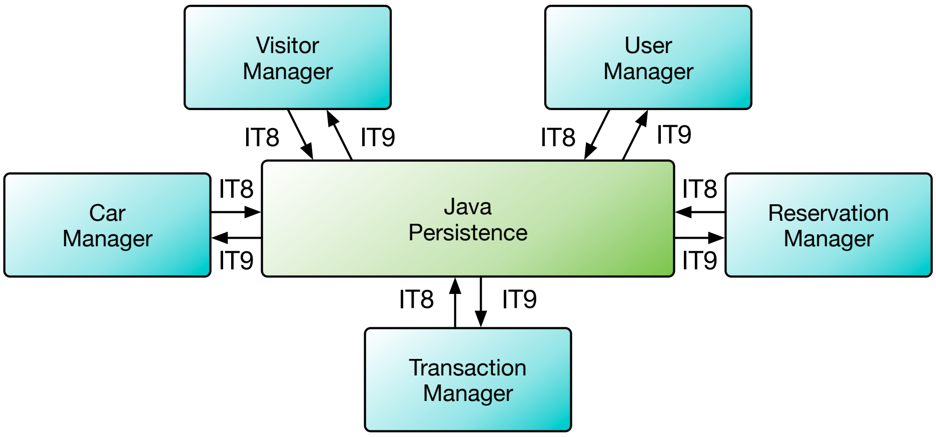

Figure 3: Business-tier subcomponents and Java Perststence connections

ID	Components	IT	TP
IT8	User Manager → Java Persistence	3.1.8	3.2.3
IT8	Visitor Manager → Java Persistence	3.1.8	3.2.3
IT8	Car Manager → Java Persistence	3.1.8	3.2.3
IT8	Reservation Manager → Java Persistence	3.1.8	3.2.3
IT8	Transaction Manager → Java Persistence	3.1.8	3.2.3
IT9	Java Persistence → User Manager	3.1.9	3.2.3
IT9	Java Persistence → Visitor Manager	3.1.9	3.2.3
IT9	Java Persistence → Car Manager	3.1.9	3.2.3
IT9	Java Persistence → Reservation Manager	3.1.9	3.2.3
IT9	Java Persistence → Transaction Manager	3.1.9	3.2.3

Table 3: Business-tier subcomponents and Java Perststence connections

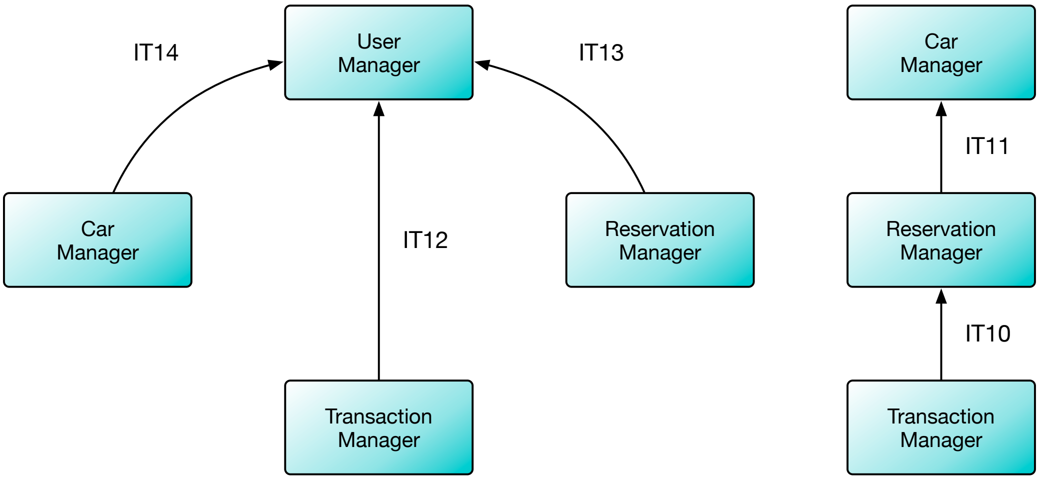

Figure 4: Business-tier subcomponents connections

ID	Components	IT	TP
IT10	Transaction Manager → Reservation Manager	3.1.10	3.2.4
IT11	Reservation Manager → Car Manager	3.1.11	3.2.4
IT12	Transaction Manager → User Manager	3.1.12	3.2.4
IT13	Reservation Manager → User Manager	3.1.13	3.2.4
IT14	Car Manager → User Manager	3.1.14	3.2.4

Table 4: Business-tier subcomponents connections

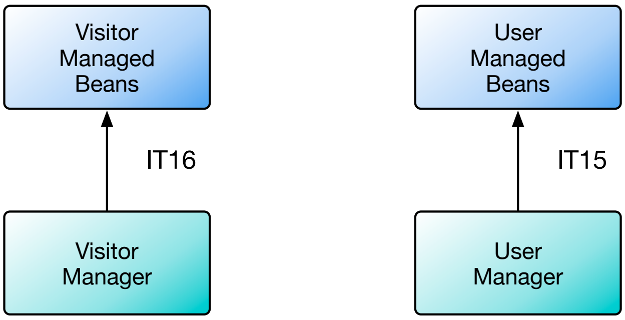

Figure 5: Managed Beans and corresponding Managers connections

ID	**Components**	**IT**	**TP**
IT15	User Manager→ User Managed Beans	3.1.15	3.2.5
IT16	Visitor Manager → Visitor Managed Beans	3.1.16	3.2.5

Table 5: Managed Beans and corresponding Managers connections

3 Individual Steps and Test Description

3.1 Test Case Specifications

3.1.1 Integration Test Case IT1

Test Case Identifier	IT1T1
Test Item	Visitor Managed Beans → Visitor Manager
Input Specifications	Create typical Visitor Managed Beans input
Output Specifications	Check if the correct methods are called in the Visitor Manager
Environmental Needs	Client driver

Table 6: Integration Test Case IT1T1

3.1.2 Integration Test Case IT2

Test Case Identifier	IT2T1
Test Item	User Managed Beans → User Manager
Input Specifications	Create typical User Managed Beans input
Output Specifications	Check if the correct methods are called in the User Manager
Environmental Needs	Client driver

Table 7: Integration Test Case IT2T1

3.1.3 Integration Test Case IT3

Test Case Identifier	IT3T1
Test Item	User Manager → Car Manager
Input Specifications	Create typical User Manager input
Output Specifications	Check if the correct methods are called in the Car Manager
Environmental Needs	IT2 succeded

Table 8: Integration Test Case IT3T1

3.1.4 Integration Test Case IT4

Test Case Identifier	IT4T1
Test Item	User Manager → Reservation Manager
Input Specifications	Create typical User Manager input
Output Specifications	Check if the correct methods are called in the Reservation Manager
Environmental Needs	IT2 succeded

Table 9: Integration Test Case IT4T1

3.1.5 Integration Test Case IT5

Test Case Identifier	IT5T1
Test Item	User Manager → Transaction Manager
Input Specifications	Create typical User Manager input
Output Specifications	Check if the correct methods are called in the Transaction Manager
Environmental Needs	IT2 succeded

Table 10: Integration Test Case IT5T1

3.1.6 Integration Test Case IT6

Test Case Identifier	IT6T1
Test Item	Car Manager → Reservation Manager
Input Specifications	Create typical Car Manager input
Output Specifications	Check if the correct methods are called in the Reservation Manager
Environmental Needs	IT3, IT2 succeded

Table 11: Integration Test Case IT6T1

3.1.7 Integration Test Case IT7

Test Case Identifier	IT7T1
Test Item	Reservation Manager → Transaction Manager
Input Specifications	Create typical Reservation Manager input
Output Specifications	Check if the correct methods are called in the Transaction Manager
Environmental Needs	IT6, IT4, IT3, IT2 succeded

Table 12: Integration Test Case IT7T1

3.1.8 Integration Test Case IT8

Test Case Identifier	IT8T1
Test Item	User Manager → Java Persistence
Input Specifications	Create typical User Manager input
Output Specifications	Check if the correct methods are called in the Persistence Module
Environmental Needs	IT2 succeded

Table 13: Integration Test Case IT8T1

Test Case Identifier	IT8T2
Test Item	Visitor Manager → Java Persistence
Input Specifications	Create typical Visitor Manager input
Output Specifications	Check if the correct methods are called in the Persistence Module
Environmental Needs	IT1 succeded

Table 14: Integration Test Case IT8T2

Test Case Identifier	IT8T3
Test Item	Car Manager → Java Persistence
Input Specifications	Create typical Car Manager input
Output Specifications	Check if the correct methods are called in the Persistence Module
Environmental Needs	IT3, IT2 succeded

Table 15: Integration Test Case IT8T3

Test Case Identifier	IT8T4
Test Item	Reservation Manager → Java Persistence
Input Specifications	Create typical Reservation Manager input
Output Specifications	Check if the correct methods are called in the Persistence Module
Environmental Needs	IT6, IT3, IT2 succeded

Table 16: Integration Test Case IT8T4

Test Case Identifier	IT8T5
Test Item	Transaction Manager → Java Persistence
Input Specifications	Create typical Transaction Manager input
Output Specifications	Check if the correct methods are called in the Persistence Module
Environmental Needs	IT7, IT6, IT4, IT3, IT2 succeded

Table 17: Integration Test Case IT8T5

3.1.9 Integration Test Case IT9

Test Case Identifier	IT9T1
Test Item	Java Persistence → User Manager
Input Specifications	Create typical Java Persistence input
Output Specifications	Check if the correct methods are called in the User Manager
Environmental Needs	Database Driver

Table 18: Integration Test Case IT9T1

Test Case Identifier	IT9T2
Test Item	Java Persistence → Visitor Manager
Input Specifications	Create typical Java Persistence input
Output Specifications	Check if the correct methods are called in the Visitor Manager
Environmental Needs	Database Driver

Table 19: Integration Test Case IT9T2

Test Case Identifier	IT9T3
Test Item	Java Persistence → Car Manager
Input Specifications	Create typical Java Persistence input
Output Specifications	Check if the correct methods are called in the Car Manager
Environmental Needs	Database Driver

Table 20: Integration Test Case IT9T3

Test Case Identifier	IT9T4
Test Item	Java Persistence → Reservation Manager
Input Specifications	Create typical Java Persistence input
Output Specifications	Check if the correct methods are called in the Reservation Manager
Environmental Needs	Database Driver

Table 21: Integration Test Case IT9T4

Test Case Identifier	IT9T5
Test Item	Java Persistence → Transaction Manager
Input Specifications	Create typical Java Persistence input
Output Specifications	Check if the correct methods are called in the Transaction Manager
Environmental Needs	Database Driver

Table 22: Integration Test Case IT9T5

3.1.10 Integration Test Case IT10

Test Case Identifier	IT10T1
Test Item	Transaction Manager → Reservation Manager
Input Specifications	Create typical Transaction Manager input
Output Specifications	Check if the correct methods are called in the Reservation Manager
Environmental Needs	IT9 succeded

Table 23: Integration Test Case IT10T1

3.1.11 Integration Test Case IT11

Test Case Identifier	IT11T1
Test Item	Reservation Manager → Car Manager
Input Specifications	Create typical Reservation Manager input
Output Specifications	Check if the correct methods are called in the Car Manager
Environmental Needs	IT9, IT10 succeded

Table 24: Integration Test Case IT11T1

3.1.12 Integration Test Case IT12

Test Case Identifier	IT12T1
Test Item	Transaction Manager → User Manager
Input Specifications	Create typical Transaction Manager input
Output Specifications	Check if the correct methods are called in the User Manager
Environmental Needs	IT9 succeded

Table 25: Integration Test Case IT12T1

3.1.13 Integration Test Case IT13

Test Case Identifier	IT13T1
Test Item	Reservation Manager → User Manager
Input Specifications	Create typical Reservation Manager input
Output Specifications	Check if the correct methods are called in the User Manager
Environmental Needs	IT10, IT9 succeded

Table 26: Integration Test Case IT13T1

3.1.14 Integration Test Case IT14

Test Case Identifier	IT14T1
Test Item	Car Manager → User Manager
Input Specifications	Create typical Car Manager input
Output Specifications	Check if the correct methods are called in the User Manager
Environmental Needs	IT9, IT10, IT11 succeded

Table 27: Integration Test Case IT14T1

3.1.15 Integration Test Case IT15

Test Case Identifier	IT15T1
Test Item	User Manager → User Managed Beans
Input Specifications	Create typical User Manager input
Output Specifications	Check if the correct methods are called in the User Managed Beans
Environmental Needs	IT9, IT12, IT13, IT14 succeded

Table 28: Integration Test Case IT15T1

3.1.16 Integration Test Case IT16

Test Case Identifier	IT16T1
Test Item	Visitor Manager → Visitor Managed Beans
Input Specifications	Create typical Visitor Manager input
Output Specifications	Check if the correct methods are called in the Visitor Managed Beans
Environmental Needs	IT9 succeded

Table 29: Integration Test Case IT16T1

3.2 Test Procedures

3.2.1 Test Procedure TP1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the User Manager and the Visitor Manager can successfully: • receive requests from User Managed Beans and Visitor Managed Beans respectively • correctly elaborate them
Procedure Steps	Execute IT1, IT2

Table 30: Test Procedure TP1

3.2.2 Test Procedure TP2

Test Procedure Identifier	TP2
Purpose	This test procedures verifies whether Car Manager, Reservation Manager and Transaction Manager can receive and handle requests from User Manager. Also, this procedure verifies whether Reservation Manager can receive and handle requests from Car Manager and Transaction Manager from Reservation Manager
Procedure Steps	Execute IT3, IT4, IT5, IT6, IT7

Table 31: Test Procedure TP2

3.2.3 Test Procedure TP3

Test Procedure Identifier	TP3
Purpose	This test procedure verifies whether the Java Persistence Module can successfully receive, handel and reply requests from: • Visitor Manager • User Manager • Car Manager • Transaction Manager • Reservation Manager
Procedure Steps	Execute IT8, IT9 in this order

Table 32: Test Procedure TP3

3.2.4 Test Procedure TP4

Test Procedure Identifier	TP4
Purpose	This test procedure verifies whether the User Manager can successfully receive and handel requests from: • Car Manager • Transaction Manager • Reservation Manager It also verifies if Reservation Manager can receive and handle requests from Transaction Manager and Car Manager from Reservation Manager
Procedure Steps	Execute IT10, IT11, IT12, IT13, IT14

Table 33: Test Procedure TP4

3.2.5 Test Procedure TP5

Test Procedure Identifier	TP5
Purpose	This test procedure verifies whether the User Managed Beans can receive and elaborates requests from User Manager and Visitor Managed Beans from Visitor Manager
Procedure Steps	Execute IT15, IT16

Table 34: Test Procedure TP5

4 Tools and Test Equipment Required

In this section of the document, we are going to identify all tools and test equipment needed to accomplish the integration and to explain why we are going to use them. In order to carry out the Integration Test, we think that the best solution is to use `Arquillian` (more info at `http://arquillian.org`). It is an integration testing used to execute test cases against the container: interaction with the system as important as the performed work. Moreover, an Arquillian test it is not complex, because it looks just like a *JUnit test*, but with some more functionalities. Another positive aspect is that its framework is compatible with JEE containers, that are the ones on which this project relies on.

Furthermore, we think that a tool such as Jmeter (more info at `http://jmeter.apache.org`) can be useful. It allows us to load test functional behavior and measure performance. It can be used to simulate a heavy load on a server, network, or object, to test its strength or to analyze overall performance under different load types. This can be a good way to verify the scalability of our application for a large number of user.

In addition to what described above, a GPS Receiver and a smarphone (or a tool to simulate them) are needed, with characteristics that respect all the requirements we have already defined in the RASD.

5 Program Stubs and Test Data Required

5.1 Program Stub

In order to carry out the testing phase for our project, we need two specific drivers that will simulate the behaviour of the client side. In particular, we need at least the Visitor Driver and the User Driver.

The Visitor Driver should simulate the calling of the following function:

- create a new user, by filling the registration form;
- log in, by inserting the proper username and password.

The User Driver should simulate the calling of the following function:

- vizualize the Personal Page;
- modify the information on his Personal Page;
- reserve a car;
- delete an already existent reservation;
- vizualize the Safe Areas near to a location;
- vizualize the Power Grid Stations near to a location;
- vizualize all its past transactions;

5.2 Test Data Required

In order to carry out the testing phase for our project, we need to populate some Database table that will simulate the real data that our application needs.

In particular, we need to populate the following tables:

- User;
- Car;
- Reservation;
- Safe Area;
- Power Grid Station;
- Transaction;

Moreover, it would be useful to have a dataset of location data, so that we can simulate inputs from from the users' smartphones and the GPS Receiver of the cars.

6 Appendix

In this section, we will give the information about the used tools, the hours of work done by the members of the group.

6.1 Used Tools

In this phase of the project, the following tools have been used:

- LaTeX and TeXMaker editor: to redact and to format this document
- Omnigraffle (`https://www.omnigroup.com/omnigraffle`): to make graphs

6.2 Working Hours

Last Name	**First Name**	**Total Hours**
Blanco	Federica	16
Casasopra	Fabiola	16