

**Software Engineering 2: “PowerEnJoy”**

**Test Plan (V. 1.0)**

Authors:

Sergio CAPRARA

Soheil GHANBARI

Erica TINTI

Milan, Italy

11/01/2017

**Table of contents**

[1. Introduction 3](#_Toc470825787)

[1.1 Revision History 3](#_Toc470825788)

[1.2 Purpose and Scope 3](#_Toc470825789)

[1.3 Definitions and Abbreviations 3](#_Toc470825790)

[1.4 Reference Documents 4](#_Toc470825791)

[2. Integration Strategy 5](#_Toc470825792)

[2.1 Entry Criteria 5](#_Toc470825793)

[2.2 Elements to be Integrated 5](#_Toc470825794)

[2.3 Integration Testing Strategy 5](#_Toc470825795)

[2.3.1 Central Application 5](#_Toc470825796)

[2.3.2 Persistence 6](#_Toc470825797)

[2.4 Sequence of Component/Function Integration 6](#_Toc470825798)

[2.4.1 Software Integration Sequence 6](#_Toc470825799)

[2.4.2 Subsystem Integration Sequence 6](#_Toc470825800)

[3. Individual Steps and Test Description 7](#_Toc470825801)

[3.1 Sample 7](#_Toc470825802)

[4. Tools and Test Equipment Required 8](#_Toc470825803)

[4.1 Sample 8](#_Toc470825804)

[4.2 Sample 8](#_Toc470825805)

[5. Program Stubs and Test Data Required 9](#_Toc470825806)

[6. Effort Spent 10](#_Toc470825807)

# Introduction

## Revision History

In this document we are providing.

## Purpose and Scope

In this document, we are providing details on how the components described in the Design Document will be tested. To ensure that the interaction between them will give the expected results, we are choosing the method to follow and we are keeping in mind that Unit Test will be done before starting the Integration Test phase.

In the following chapters you will find detailed descriptions of the tests and the name of the tools to be used.

## Definitions and Abbreviations

* **User:** the person registered to the system and allowed to access to its functions.
* **Operator:** a person with technical skills, that fixes car issues.
* **App:** short term used to define a mobile application.
* **Power Plug:** a column with one or more electricity socket where it is possible to charge the car.
* **Safe Area** (or Parking Area): a parking area with parking shared with all the other divers and not especially reserved to PowerEnjoy.
* **Special Parking Area** (or Power Station): a parking area reserved exclusively to PowerEnjoy cars where, for each parking space there is a Power Plug where it is possible to charge a car.
* **Car:** PowerEnjoy car.
* **Reservation:** the relation between a user and a car, that allows the user to start using the car. The reservation guarantees that no one else can reserve and use the reserved car till the end of the rental.
* **DB:** database, the collection of system data.
* **GUI:** Graphic User Interface, the interface that allows the user to interact with the system.
* **RASD:** Requirements Analysis and Specifications Document.
* **DAO:** Data Access Object.
* **DTO:** Data Transfer Object.
* **MVC:** Model-View-Controller, the pattern used for the development.
* **UX:** user experience.
* **GEB:** Green e-Box.

## Reference Documents

The documents used as a reference to provide the design document are:

* Assignments AA 2016-2017.pdf
* Sample Design Deliverable Discussed on Nov. 2.pdf
* IEEE Standard for IT – System Design – Software Design Description
* Structure of the design document
* Paper on the green move project.pdf
* Second paper on the green move project.pdf

# Integration Strategy

## Entry Criteria

Before starting the Integration testing, we have to ensure that unit tests have been done on all the components and classes of the system.

## Elements to be Integrated



Our system is composed by many components, that we can distinguish in two level of granularity.

The higher level of components of our system that needs to be integrated are:

* User Application;
* Operator Application;
* Central Application, containing the main system logic;
* Model
* Database.

These higher level components are composed by some lower level components that needs to be integrated too. More in detail, the component Central Application is composed by Authentication, WebService, Maintenance Controller, Reservation Controller.

Since the Calculation Controller component has a passive role because it is just “called” by the others components and never calls others components, we decided that unit test of this component are enough, otherwise unit test and integration test for Calculation Controller would be the same.The main application interacts with the database to get all information concerning users, operators, cars, parking areas and special parking areas. Most of the interactions concern the request or the update of car information, such as its position before reserving it or after parking it.

## Integration Testing Strategy

We decided to use a bottom-up approach to test the software. The choice is based on the fact this method guarantees a better result while testing. In fact, we think that the use of drivers, when needed, would be better than working with stubs. Also, this will prevent long working time for making stubs.

As already discussed, the software has many critical points, but we guess that the bottom-up strategy will be a safer method and software faults will be easily found and corrected.

### **Central Application**

The controller has components specialised in different operations. These are:

* the Web Service, that acts as a dispatcher for the incoming requests from user applications, checks the correct authentication and gets the results from the other controllers;
* the Authentication, which is used for the validation of the user login information;

The DAOs exchange messages directly with the Database, while the DataInterface lets the model communicate with the DataService.

### **Persistence**

The persistence contains the database, where all the data of the system is stored, and the DataService component.

## Sequence of Component/Function Integration

The interfaces we use are described in the following paragraphs. These play an important role because they grant the communication between the components of our system.

### **Software Integration Sequence**

SORTING NEEDED?

|  |  |  |
| --- | --- | --- |
| **ID** | **Integration Test** | **Paragraph** |
| **I1** | DataService → CarDAO  DataService → SParkingAreaDAO |  |
| **I2** | Database → UserDAO  Database → OperatorDAO  Database → CarDAO  Database → ParkingAreaDAO  Database → SParkingAreaDAO |  |
| **I3** | CarDAO → MaintenanceController  ParkingAreaDAO → MaintenanceController  SParkingAreaDAO → MaintenanceController |  |
| **I4** | CarDAO → ReservationController  ParkingAreaDAO → ReservationController  SParkingAreaDAO → ReservationController |  |
| **I5** | UserDAO → Authentication  OperatorDAO → Authentication |  |
| **I6** | ReservationController → WebService |  |
| **I7** | MaintenanceController → WebService |  |
| **I8** | Authentication → WebService |  |
| **I9** | WebService → UserAppController  WebService → OperatorAppController |  |

### **Subsystem Integration Sequence**

This interface provides the methods to be implemented in the DAO components. Default methods for the DAOs are the following:

# Individual Steps and Test Description

## Sample

When the user has accessed the app and has logged in, he is automatically redirected to the map view, displaying the available cars. The function that is used to prepare the map view of the area around the user is implemented as follows:

# Tools and Test Equipment Required

## Sample

The interface of the mobile applications has already been presented on the RASD, but we wish to add some other screens that we decided to add.

## Sample

The diagram shows how user actions are performed and the sequence of the navigation between the screens.

# Program Stubs and Test Data Required

Here we present the components that are involved in the fulfilment of the goals presented in the RASD.

# Effort Spent

For the document, each one of us has worked around 32 hours.