

PROJECT for

SOFTWARE ENGINEERING 2

**Testing Document**

Politecnico di Milano

A.A. 2016-2017

Prof.ssa Elisabetta Di Nitto

Students:

Diego Gaboardi

Giorgio Giardini

Riccardo Giol

INDEX

1 INTRODUCTION

* 1. REVISION HISTORY
  2. PURPOSE AND SCOPE

This document is the Integration Test Plan Document for PowerEnjoy.

The main purpose is to describe in a clear and comprehensive way how we plan to accomplish the integration test, in order to integrate all different subsystems that make up our application, allowing in this way a correct and consistent realization of the whole project.

In particular, we’re going to provide:

* A global vision on which are principal subsystem and subcomponents that we want to integrate and test
* A description of the approach that we will follow for test implementation
* The effective order for components and subsystems integration, with a detailed description of each integration step, including the expected output for each input data
* Some measures of expected components performance in order to completely satisfy the requirements
* A list of all the tools and environments that will have to be used during test acivity
  1. DEFINITIONS, ACRONYMS, ABBREVIATIONS
  2. REFERENCE DOCUMENTS

1. INTEGRATION STRATEGY
   1. ENTRY CRITERIA

In this paragraph, we are going to describe all the conditions necessary for starting the Integration Testing of our project.

First of all the Requirement Analysis and Specifications Document (RASD) must be written in order to have all the functional and non-functional requirements already defined between the customers and the developers. At the same time also the Design Document (DD) is required because it is necessary to be conscious about the components of the system and their interactions.

When the Integration Testing begins we must have already started the Coding and Unit Test phase. However these two phases can be done partially in parallel, progressively integrating modules in subsystems as soon as they are implemented.

For this reason we give an estimation of the minimum percentage of completion of the components of our system required in order to start this phase. This values are calculated reflecting the order in which components are integrated.

* 100% for the DataController component;
* At least 95% for the ClientController and CarController components;
* At least 80% for the CourseController component;
* At least 60% for the ReservationController component;
* At least 40% for the AssistantController component;
* At least 40% for the client-side components.

In this way we evaluated that the two phases would be done concurrently in the most efficient way.

* 1. ELEMENTS TO BE INTEGRATED

In this paragraph, we are going to list all the components that will be incrementally integrated during the Integration Testing.

For this purpose we took into account the Component Diagram defined in the Design Document in which we showed a high-level representation of the components of our system focusing on their relationships. However, some components will be decomposed in sub-components in order to have a model of our system with a lower level of abstraction. This will help us to define the best strategy for integrating the components in subsystems.

The components are the following:

* Data Controller: it is the component that interacts with the DBMS in order to obtain the data stored or to save new tuples.
* ClientController: it is the component that handles the client accounts and so it is decomposed in several sub-components that carry out the required task: ClientRegistrationController, CientLoginController and ClientProfileController. At the same time it also manages the current state of the client and so it is composed also by a ClientStateController.
* CarController: it is the component which handles all the information about cars: position, state and battery.
* CourseController: it is the component which manages the courses. This is one of the main functionality offered by our system and so it has to interact with other components such as the ParkingStationController, the SafeAreaController and the ChargeCalculator.
* ReservationController: this component is responsible for client reservations and interacts with almost the same components of the CourseController.
* AssistantController: it manages assistant accounts and so it has subcomponents similar to the ClientController component such as the AssistantRegistrationController, the AssistantLoginController and the AssistantProfileController.
* Some commercial components which have been already developed such as the DBMS and the PaymentSystem.
* The components on the client-side of our system and so the ClientApplicationController, the AssistantApplicationController and the CarApplicationController.
  1. INTEGRATION TESTING STRATEGY

In this paragraph we are going to explain the strategy that we will use for performing the Integration Testing describing the order in which components are considered.

In particular our main goal is to parallelize as much as possible this phase with the Coding phase and at the same discover bugs early in the most critical components of our system. For this reason we will partially adopt a bottom-up approach and so we will start integrating components that depends only on already developed and integrated components. This approach will be combined with a critical module testing and so we will start by integrating the most important sub-systems of components.

This is the order chosen:

* The DataController with the DBMS because these components are used in almost all the tasks. The DBMS is an external and commercial component and so we considered it already developed.
* The subsystem about clients and so the ClientController and its sub-components: the ClientRegistrationController, the CientLoginController and the ClientProfileController.
* The subsystem about the cars.
* The components relative to courses. Even if this is the most critical subsystem we decided to integrate it at this point in order to have all the components necessary for this task already developed and integrated.
* The components relative to reservations. They are considered now for the same reasons of the previous point.
* The subsystem about assistants and so the AssistantController and its sub-components: the AssistantRegistrationController, the AssistantLoginController and the AssistantProfileController.
  1. SEQUENCE OF COMPONENT / FUNCTION INTEGRATION

1. INDIVIDUAL STEP - TEST DESCRIPTION
2. PERFORMANCE ANALYSIS

With the performance analysis we evaluate in which conditions and with which tools the system work properly. A complete analysis can be performed during the implementation of the infrastructure, monitoring how the system responds with the structures used. However, is useful perform a preliminary analysis for the infrastructures required from the performances requested.

For the correct running of the mobile application, the smartphone has to provide at least:

* The correct tracking through GPS;
* Camera with 5Mpx;
* 3,5 MB of memory for the app;
* 80 MB of RAM to be execute;
* CPU with 1GHz in one core.

All these constraints are approximate, and some more close to real values can be detected in the future, with the implementation of the infrastructure.

The application will be developed with the framework Apache-Cordova for all the platforms. Than it is tested with a specific SDK for different smartphones.

1. TOOLS - TEST EQUIPMENT REQUIRED

The tests are all executed with some automatized tools, that decrease the writing time for the test and simplify and get more readable the solutions. It is even easier to find the error in the code, thanks to the debugging offer by the IDE Netbean.

6 STUBS, DRIVERS AND TEST DATA