Version 2.1

Software Engineering Project

Integration Test Plan

POLITECNICO
MILANO 1863

Vianney Payelle - Rémi Rigal - Noëlie Ramuzat

Revision Notice

Versions	Date	Description	Modifications
V1.1	03/01/2017	Creation of the document	Add of the Introduction part
V1.2	06/01/2017	Adding of the Integration Strategy part	Add of the Integration Strategy part
V1.3	15/01/2017	Validation of the document	Add of the : - Test Description part - Tools and Equipment part
V2.1	08/02/2017	Correction of the document	Correction of the Test description part

Table 1 : Testing Document versions

Table of contents

List of tables

List of figures

1. Introduction

1.1. Purpose

The Integration Test Plan Document provides a dynamic technique to verify and validate the activities and software artefacts of a project.

Its aim is to find and show the presence of bug in the project. In this document is used systematic testing which uses the structure of the software artefacts and some information on the system's behaviour…

1.2. Scope

The scope of the project PowerEnjoy, which is a service based on mobile application, is to manage, design, build, and implement a service aimed at facilitating public transportation. The application provides to its target, the client, a way to research an electric car near a position, reserve it and pick it for a ride. At the end the application sends the ride's bill to the client. The PowerEnjoy application needs the client to be registered in its database before he can reserve a car, for security and payment reasons (credential information, driver license, and identity card). When the client logs in, the mobile application allows him to reserve a car around an address or his GPS position. Then it provides him details about his reservation on the main page. The application also allows the client to cancel his reservation, unlock the reserved car when he is near it, and access his account details to modify it.

After the ride the application locks the car and sends an email to the client with the bill of the ride. The mobile application can moreover give discounts and charges in function of the client's ride such as sharing the car or plug it in power grid station. The PowerEnjoy application is built in order to ensure an easy and clear reservation service as well as an effective data collecting and saving. This refers the PowerEnjoy application simplifies the customer's uses, optimises the time to reserve a car and adjusts the price of the ride according to the driver.

1.3. Definitions, Acronyms, Abbreviations

Name	Definition
RASD	Requirements Analysis and Specifications Document
DD	Design Document
BCE	Business Controller Entity
PaaS	Platform as a Service : permit to control a software deployment
DBSM	Database Management System
API	Application Programming Interface.
SMS	Short message service: notification sent to a mobile phone SMS gateway is needed to use it
OS	Operating Systems

Table 2 : Glossary of the Testing Document

1.4. Reference documents

Name	Publication Date	Authors	Contents
Assignment AA 2016-2017 Software Engineering 2	14/10/2016	Elisabetta di Nitto	Project goal, schedule and rules
RASD	16/11/2016	Rémi Rigal Vianney Payelle Noëlie Ramuzat	Requirements Analysis and Specifications Document of the project
DD	15/12/2016	Rémi Rigal Vianney Payelle Noëlie Ramuzat	Design Document of the project

Table 3 : Reference documents used in the Testing Document

1.5. Used Tools

Name	Use
Github & SourceTree	Control the document versions
Edraw Max 8.4	Create the Architecture models
Adobe Acrobat Reader DC	Create the Integration document PDF

Table 4 : Description of the tools used to create the Testing document

1.6. Document structure

In the following parts of the document are described the main characteristics of the integration testing plan.

- Integration Strategy

This section introduces the elements to be tested and how they will be integrated, describing moreover the other criteria related to the integration.

- Individual Steps and Test Description

This part describes the type of tests that will be used to verify that the elements integrated at each step perform as expected.

- Tools and Test Equipment Required

In this section are presented all the testing tools needed to accomplish the integration.

- Program Stubs and Test Data Required

This part identifies the program stubs and special test data that are required for each integration step.

Finally the hours of work repartition follows this section.

2. Integration Strategy

2.1. Entry Criteria

In this paragraph are presented the documents and unit tests that must have been delivered or done before making the integration test. The following documents are needed to ensure that the test process has visibility within the overall project and that the test tasks are started at the appropriate time:

- The Requirement Analysis and Specifications Document to state the requirements of the system
- The Design Document to understand the software architecture of the project

The following functions have to be unit tested with specific input to find the problems early and facilitate the changes of them by verifying the connexion between:

- The Database and the DriverManager, by making a unit test on the "verifyDriver(String email)" function and that it can be modify with a test on the "addCar(String id, String location, Int battery, Bool available)" function (linking the CarManager and the Database).
- The Mobile Application and the Router, by testing the "verifyRegistrationUser(String[] data)" function the "verifyLogin(String email, Sting password)" function and the "displayMainPage()" one.
- The ReservationController and the NotificationHelper, by checking the "sendSMS&MailConfirm()" function.
- The Car and the CarManager, by testing the "unlockCar(Car car)" function (linking the CarManager to the Car)
- The Router and the ReservationController/ResearchController/DriverController/LoginController with the functions "transfertRequest(args)" with the appropriate arguments.

From a components point of view, before testing the program should contains an amount of code for each one:

- The Database has to be ready for an architectural point of view, it should lack only some data, so a total completion of 90% of the code
- The DriverManager and LoginManager have to present the possibilities to register, login and reserve a car so a total completion of 70% of the code

- The ResearchManager and ReservationManager have to be nearly achieved so a total completion of 80% of the code
- The RideManager and NotificationHelper are less critical and can be only achieved partially so a total completion of 50% of the code

Before delivering the product, all these components have to present a minimum of 95% of completion. This list presents a completion priority that corresponds to a critical order of the components; it can explain the following choices made to integrate the sub-systems during the tests.

2.2. Elements to be Integrated

In this part are identified the components that have to be integrated to perform the testing plan of the system. Based on the Subsystem and Component View diagrams of the Design Document, the selected components are the following:

- The RideManager
- The CarManager
- The ReservationController
- The ResearchController
- The DriverController
- The LoginController
- The NotificationHelper (Manager)
- The DataAccessManager : it is the link between the Database Management System and the other components

These ones have been chosen because they are the code modules that manage the entire program. They are in charge of the requests transfer between each other's or external items (NotificationHelper to the three Gateways) to create the model.

The PowerEnjoy system is based on these interactions between these high-level components which represent the sub-systems of the program and hold a unique functionality (DriverManager: History, Account, Credentials, and ReservationManager: Car, Ride).

According to the dependencies between the components, some sets of sub-systems can be created to identify clearly the linked components to be integrated in the test.

The Driver Systems composed of the DriverManager, the LoginManager and the NotificationHelper sub-systems.

The Car Systems composed of the CarManager, ResearchManager, ReservationManager, RideManager and NotificationHelper sub-systems.

It can be noticed that the NotificationHelper and the DataAccessManager take part of both sub-systems set due to the dependencies of the two sub-systems to them.

Other external components have to be taken into account during the integration tests, but don't have to be tested (because they already exist):

- The Mail/SMS/Push Gateways, this component is linked to the NotificationHelper, it permits to send the sms/email/notifications to the driver
- The CarDataService, this component is linked to the CarManager, it transfers the data of the car to the PowerEnjoy system
- The BankTransactionService, this component is linked to the DriverManager, it permits to perform the payments
- The Database Manager System, linked to the DataAccessManager, it manages the requests addressed to the Database

These externals components represent the highest level subsystems and because they already have been tested will not be part of the integration tests.

Finally these are the components to be integrated group by set of sub-systems. In the following diagram, the arrows represent the dependencies of each component to the others. For instance C1 - - - > C2 means that the component C1 is dependent of the component C2.

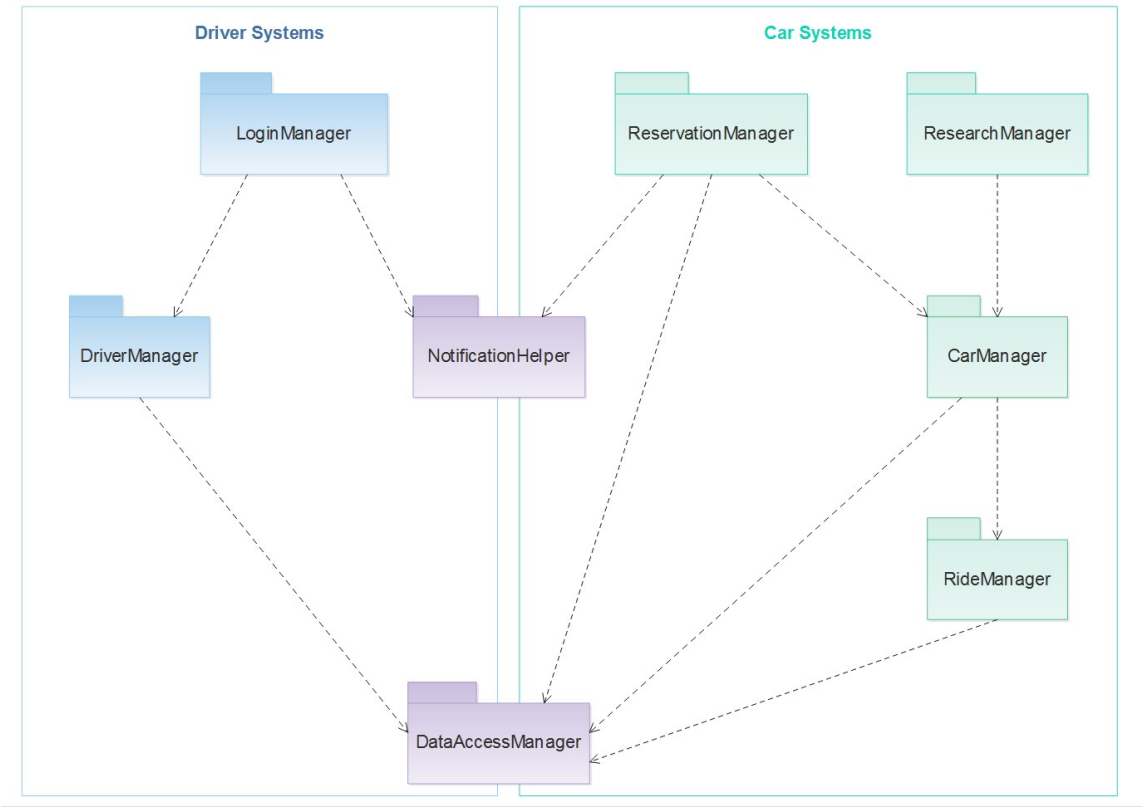

Figure 1: Component Dependencies

2.3. Integration Testing Strategy

The integration strategy we will use for our system is a bottom-up based strategy. Thus, the integration of the whole software will start with standalone components or components that depend on third party components such as DBMS, the notification systems for mails and SMS and mapping service. The integration will then continue until reaching the component of the highest level. This strategy will maximize the efficiency and the parallelism of the software development.

In our case, the most critical components are the ones interfacing with hardware such as the one managing the remote unlocking of the cars or dealing with inboard sensors. Because they are critical, they will be implemented first in order to be able to detect bugs or malfunctions as early as possible.

In fact, the bottom-up strategy allows earlier testing than the top-down one making the integration process easier to follow and control. Indeed, the testing team can start performing integration testing as soon as a component has been developed, thus providing precise information about the behaviour of specific components in several states.

2.4. Component Integration Sequence

DataAccessManager

Most of the components of the system rely on the database to work. Therefore, the DataAccessManager will be integrated first using the third party Database Management System.

NotificationHelper

The NotificationHelper is used by the two sub-systems for the driver and the car management; this one will be done next using the different third party gateways for push messages, text messages and mails.

2.4.1. Driver Management Systems

DriverManager

The DriverManager is the essential component of the driver sub-system as he is the one handling all the personal information of all the clients. It depends on the third party BankTransactionService as it is the one managing the payments from the drivers. It requires also the DataAccessManager to access the data of the drivers in the database.

LoginManager

The LoginManager is responsible for opening and attributing sessions for all the drivers that want to log into to the system or that want to register. This component requires the DriverManager to check the credentials of the drivers, the DataAccessManager to validate and save logs for sessions and the NotificationHelper if sending a email of a SMS is needed, during the registration process for example.

2.4.2. Car Management Systems

RideManager

The RideManager is the component managing the rides, starting from the remote unlocking of the car to the parking of it. He must be able to access and modify data so it depends on the DataAccessManager.

CarManager

The CarManager is the key component of the car management sub-system, it manages all the data concerning the car. Indeed, it needs the CarDataService that provides data from the sensors of the car such as the position, the number of person inside or the battery level. It allows the locking and the unlocking of the car, when used, the CarManager can then start or end a ride through the RideManager he depends on. This component requires also the DataAccessManager to update the data in the database.

ResearchManager

The ResearchManager is used whenever a driver looks for an available car, the needed data are fetched from the DataAccessManager and the CarManager.

ReservationManager

The ReservationManager handles the reservation from the drivers, starting when a car is booked and ending when the reservation has been cancelled or paid. It depends on the NotificationHelper to be able to notify the driver on the current status of his reservation. And it requires the DataAccessManager to updates reservation data.

The diagram below sums up the integration flow of the entire system. An arrow from a component C1 and pointing at another component C2 means that the components C1 needs to be integrated before the component C2.

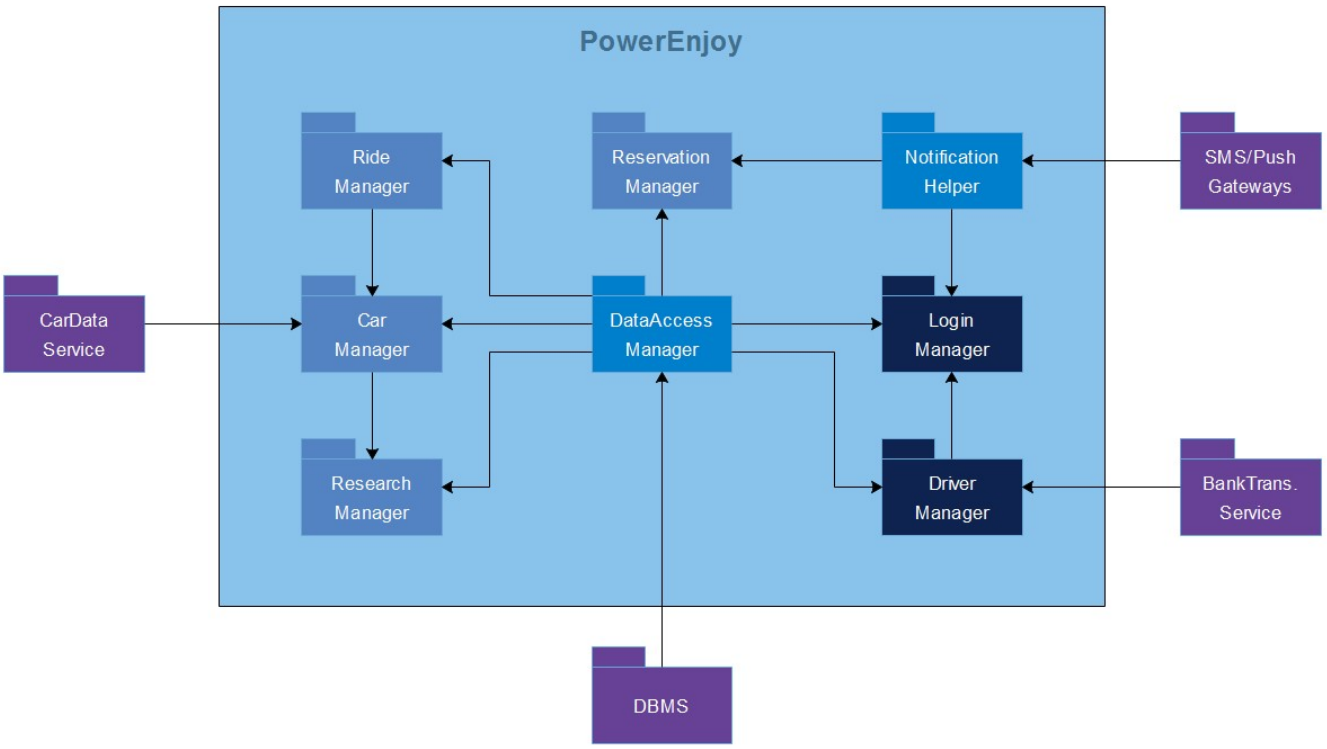

Figure 2: Component Integration Diagram

2.5. Environmental Needs

Set of elements and recommendation needed to perform the tests and can be specific to our system:
Simulation of the element those are external to the software:

- Behaviour test: what is happening in case of missing data or wrong data received/send
- Simulator of car behaviour: simulate information exchanged between the car and the system
- Simulator of user behaviour: simulate the behaviour of the user through the application

These simulations will be used before the beta test involving a sample of cars and users. It can also be used to populate the database.

The data provided by the simulators should have the same form as the real ones. Early test involving few data and element will be run in a private secure network to prevent the access of external users and security breach.

2.6. Responsibilities

In this part are presented the different managers who will work on the several integration test parts.

They will be split in groups, according to their domain of expertise and responsibilities, on the following subject:

- The Network implementation part: The group is in charge of the development of an internal network to run secure test
- The Security of system part: The group is in charge of running penetration test on the system to detect breach before testing on open network
- The Simulators development part: The group is in charge of the development of users and cars simulators to minimize the time needed on real condition test
- The General behaviour of the system part: The group is in charge of checking the behaviour of the system during the integrating of the different parts
- The System integration part: The group is in charge of the general manager integration plan creation
- The Unit testing part: The group is in charge of the global management of unit testing

3. Individual Steps and Test Description

3.1. Integration Test

3.1.1. Case I1

Test Case identifier	**I1T1**
Items tested	DBMS → DataAccess Manager
Input specifications	DBMS output
Pass criteria	-The DataAccess Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	DBMS driver

3.1.2. Case I2

Test Case identifier	**I2T1**
Items tested	SMS/Push Gateways → Notification Helper
Input specifications	SMS/Push Gateways output
Pass criteria	-The Notification Helper check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	SMS/Push Gateways driver

3.1.3. Case I3

Test Case identifier	**I3T1**
Items tested	DataAccess Manager → Ride Manager
Input specifications	
Pass criteria	-The Ride Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

3.1.4. Case I4

Test Case identifier	**I4T1**
Items tested	CarData Service → Car Manager
Input specifications	
Pass criteria	-The Car Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	CarData Service driver

Test Case identifier	**I4T2**
Items tested	DataAccess Manager → Car Manager
Input specifications	
Pass criteria	-The Car Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

Test Case identifier	**I4T3**
Items tested	Ride Manager → Car Manager
Input specifications	
Pass criteria	-The Car Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I2

3.1.5. Case I5

Test Case identifier	**I5T1**
Items tested	DataAccess Manager → Research Manager
Input specifications	
Pass criteria	-The Research Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

Test Case identifier	I5T2
Items tested	Car Manager → Research Manager
Input specifications	
Pass criteria	-The Research Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I3 I4 I5

3.1.6. Case I6

Test Case identifier	I6T1
Items tested	DataAccess Manager → Reservation Manager
Input specifications	
Pass criteria	-The Reservation Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

Test Case identifier	I6T2
Items tested	Notification Helper → Reservation Manager
Input specifications	
Pass criteria	-The Reservation Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I2

3.1.7. Case I7

Test Case identifier	I7T1
Items tested	BankTransfer Service → Driver Manager
Input specifications	
Pass criteria	-The Driver Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	BankTransfer Service driver

Test Case identifier	I7T2
Items tested	DataAccess Manager → Driver Manager
Input specifications	
Pass criteria	-The Driver Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

3.1.8. Case I8

Test Case identifier	I8T1
Items tested	DataAccess Manager → Login Manager
Input specifications	
Pass criteria	-The Login Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I1

Test Case identifier	I8T2
Items tested	NotificationHelper →Login Manager
Input specifications	
Pass criteria	-The Login Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I2

Test Case identifier	I8T3
Items tested	Driver Manager → Login Manager
Input specifications	
Pass criteria	-The Login Manager check that the data coming in and out are valid
	-If the data coming in are valid, check that it calls the right methods
	-If the data are not valid, signal an error in the process and check that the error has been caught
Environmental needs	I7

3.2. Test procedure

3.2.1. TP1

Test Procedure identifier	TP1
Item to test	DataAccess Manager
Purpose	This test will check if the DataAcess Manager can:
	-perform modification and read in the database
	-properly communicate with the other elements of the system
Steps	I1

3.2.2. TP2

Test Procedure identifier	TP2
Item to test	Notification Helper
Purpose	This test will check if the Notification Helper can:
	-notify the driver of the different event
	-properly use the different elements of communication with the driver
Steps	I2

3.2.3. TP3

Test Procedure identifier	TP3
Item to test	Car Management System
Purpose	This test will check if the Car Management System :
	-know the information about rides and reservations
	-know the current informations about every single car
	-can use those information to signal problems
	-can share those information with the persons in charge of the cars
Steps	I1 → I3, I1-I2 → I6, I1-I3 → I4, I1-I4→ I5

3.2.4. TP4

Test Procedure identifier	TP4
Item to test	Driver Management System
Purpose	This test will check if the Driver Management System :
	-know the current informations about every single driver
	-can use those information to signal problems
	-can share those information with the persons in charge of the users
	-can communicate with the Bank Transfer Service
Steps	I1→ I7, I1-I2-I7→ I8

4. Tools and Test Equipment Required

Identify all tools and test equipment needed to accomplish the integration. Refer to the tools presented during the lectures. Explain why and how you are going to use them. Note that you may also use manual testing for some part.
Consider manual testing as one of the possible tools you have available.

4.1. Tools

To perform all the tests on the several components of the PowerEnjoy system, it is necessary to use some appropriate testing tools.

Concerning the unit testing that have to be done before the integration plan, the Mockito test tool will be used. These tests have to be fast and cover small tasks, by using mocking it is possible to abstract the dependencies and have predictable results to ensure that characteristics.

Moreover the mocking permits to verify the interactions between items and provides a framework to do so. That's why the Mockito test tool will also be used to test the business logic of the components which run in the JEE runtime environment.

Mockito will verify the interactions between the components by creating mocks; it will permit to test the results (data, exceptions and errors) but also to simulate external components like the Notification Gateways.

Still in the purpose of testing the business logic, the Arquillian testing tool will be used to check the interactions with the system according to the dependencies and the transaction controls. The tests will be performed against a Java container to ensure that the components interactions are correctly happening.

Moreover, using test archive to isolate the classes and resources it will be easier to focus on the classes/components to test. The Arquillian tool will be used in particular to monitor that the dependencies' injections are respected and to verify that the Database connections are well managed.

Concerning the tests of the functional behaviour and measure of the performance the Jmeter testing tool will be used. The purpose will be to simulate quite heavy load on the server to analyse its strength.

For instance some simulations will introduce a certain number of drivers that will try to login into the server at the same time, with then different requests to be achieved by the server.

For the concern of the mobile devices, the application must use a reasonable amount of resources (CPU + RAM) because of the limited material resources of smartphones. For that purpose, a set of monitoring tools provided by some development platforms, such as Android Studio and XCode Studio, will be used.

Finally some part of the integration testing plan requires the use of manual testing to be performed efficiently. For instance the test of the mobile interface (clicking on buttons...), the general application behaviour and the creation of the test data need manual testing operations.

4.2. Test Equipment

For each integration test is needed an appropriate testing environment, with particular equipment. The PowerEnjoy system is composed of three testing areas whose characteristics have to be defined to match the real conditions.

The first part defined the mobile devices that are required to respect the testing environment. It is needed a smartphone that runs even on an Android platform or an iOs one. For each ones, all the possible range of display size must be tested to ensure that the application’s display and resolution works correctly.

The second part concerns the sort of cars that have to be used. It must be cars with navigable screens that can be programed to communicate with the PowerEnjoy system.

The final part is about the cloud infrastructure that will be use to deploy the business logic components. A Platform as a Service (PaaS) will be use. This one will permit to control the software deployment (as a public cloud) with minimal configuration options, with provided networks, servers, storage, OS, 'middleware', database and other services.

The PaaS chosen is the CouldFoundry Foundation because it is open source and respects the same operating environment as the one on which the PowerEnjoy system runs. The CouldFoundry PaaS is a Linux Foundation Collaborative Project.

The list following describes the services and software components provided by the CouldFoundry PaaS:

- Data Storage: MySQL, PostgreSQL, MongoDB...
- Mobile: API Gateway, Data Sync, Push Notifications
- Messaging: Pivotal RabbitMQ
- Development: CloudBees Jenkins
- Source code : Apache License 2.0
- Runtimes & Frameworks : Java 6, Java 7, Java 8 (Spring Framework 3.x, 4.x)
- Application Server : Tomcat or JBOSS

5. Test Data Required

To perform the entire tests that have been described in the whole document the following data are needed:

A list of unregistered drivers who can have valid or invalid personal information. This list will be used to test the DriverManager, the LoginManager and the NotificationHelper components. The set of data must contain drivers with the following problems:

- Null object
- No arguments
- Invalid driving license
- Invalid email address
- Invalid mobile phone number

A list of cars with valid and invalid characteristics to test the CarManager and the RideManager components. This set of data has to contain cars with these problems:

- Null object
- No arguments
- Invalid battery level
- Invalid location (in the sea for instance)
- Invalid number of places

A list of researches which will have valid or invalid fields. This list will be used to test the ResearchManager component. It must contain researches with the following problems:

- Null object
- No arguments
- Invalid address location
- Invalid maximum distance

A list of reservations with valid and invalid characteristics to test the ReservationManager component. This set of data has to contain reservations with these problems:

- Null object
- No arguments
- Invalid car id
- Invalid driver email

It is to notice that the safe areas don't have to be tested because they are pre-defined in the management system, which already exist and tested.

6. Hours of work

6.1. Vianney Payelle

- 13/01/17: 1h
- 14/01/17: 2h
- 15/01/17: 5h

6.2. Rémi Rigal

- 13/01/17: 2h
- 14/01/17: 4h
- 15/01/17: 4h

6.3. Noëlie Ramuzat

- 11/01/17: 30 min
- 12/01/17: 4h
- 13/01/17: 2h
- 14/01/17: 1h
- 15/01/17: 30min