POLITECNICO
MILANO 1863

Integration Test Plan Document

Guido Muscioni (mat. 876151)
Marco Orbelli (mat. 876649)
Paola Marchesini (mat. 876541)

Figure 1: New brand logo.

Contents

1 Introduction

1.1 Purpose

This document is written based on DD document, its aim is to define the testing phase of the system. It also contains the description of the tests about the components described in the Section 2 of the document mention before. In particular, the tests will focus on the components system interfaces in order to integrate all the functions for the correct execution of the new PowerEnjoy system.
All the developers that take part in the testing and integration testing phases have to read and keep in mind the contents of that document. <TODO: verificare la correttezza di quest,ultima frase.>
The reading of that document is also suggested to all the developers of the systems, as they can focus on the critical system components.

1.1.1 Scope

The scope of that system, is defined in the RASD document, then, here there is a copy of the paragraph:

> *The aim of the project is to develop a digital system for an electric car-sharing company, that is called PowerEnjoy. There are not previous system so that document will describe the requirements that deal with all the system that the customer wants.*
> *The users of PowerEnjoy system are:*
>
> - *Client;*
> - *Employee.*
>
> *Both of these users are going to have unique credentials in order to use the system, so they must be registered.*
> *The clients will use this system to reserve and use cars of PowerEnjoy company.*
> *The employees will use that system in order to know what cars need help.*

1.2 List of Definitions and Abbreviations

<TODO:Completezza, definizioni delle percentuali fra scrittura del codice e Junit test su quel codice.>
<TODO: general system controller>

1.3 List of Reference Document

- Specification document: Assignments AA 2016-2017;
- Examples documents:

  - Sample Integration Test Plan Document.

- PayPal documentation (https://developer.paypal.com/docs/ and https://www.paypal.com/it/webapps/mpp/merchant);
- Jersey framework documentation (https://jersey.java.net/documentation/latest/user-guide.html);
- PhoneGap documentation (http://docs.phonegap.com/);
- General API documentation of mobile OS.

2 Integration Strategy

2.1 Entry Criteria

In order to begin the integration testing phase, RASD and DD documents must be approved by the company. Therefore these documents must not be modified with addiotional considerable functionalities or with relevant architectural changes. These facts are mentioned in order to try to avoid further critical alterations which can result in architectural degradationa (drift and erosion).
All what is mentioned before is used to have a complete view of the interactions between PowerEnjoy components, following the top-down methodology, used to build the Design Document.
Each component can be integrated only if it is, at least, 90% complete.
The test phase will begin according to this completeness percentage, defined in section 1.2:

- 100% for the completeness of the DataBase source, and 90% for the Data Access interface;
- 80% for the general system controller;
- 80% for the Acces Interface of the system.

2.2 Elements to be Integrated

2.3 Integration Testing Strategy

The testing phase will follow the bottom-up approach, however the system is been designed keeping in mind a top-dowm view. In actual fact the design decisions, took in the Design Document, have been made in order to make the integration phase, and also integration testing phase, easier <TODO: marco scrivila te>.
Following the bottom-up approach, once a component is complete, according to the definition of completeness in section 2.1, it can be tested, using its own driver, imediately and independently from other system components (the definition and decription of the drivers can be find in section 6.1).
Using bottom-up pattern, the test will be make on the real components that will be released with the first version of the system. Futhermore, the bugs and the errors will be find as soon as possible in order to reduce their propagation in the system. However, at the same time, the top-down approach used in design phase make the result of the system nicer than take into consideration only a general bottom-up approach, which it will be result in a bit rough outcome.
In the integration phase, the tests of commercial components, that have been used in the system, are not required, as it is supposed that the producer companies have already tested them. However the tests are required for the integration between PowerEnjoy system and these type of components.

2.4 Sequence of Component/Function Integration

In this section it is described how the different components of our system has been integrated. As a notation, it is used an arrow, which mean that a component X (starting point) needs a component Y (head of the arrow).

2.4.1 Software Integration Sequence

Following, it is explained in details how the bottom-up approch is applied to the Power Enjoy system developed. In particular, further the basic approch mentioned before, the components of the system are also been grouped according to the functionalities they provide and finally integrated together.

Data Access Interface The first elements that have to be integrated are the DataBase Source and the Data Acces Interface. The DataBase, as mentioned before in , is complete, in order to test the other components of the system using real data to perform queries and to manage a complex system as close as possible to reality.

System Controller Once made the DB accessible, it is necessary to integrate external Payment Gateway with the Payment Controller. In this way, when the next component will be integrated with the Payment component, we can test early their interaction.
>>IMMAGINE DEI DUE MODULI PAYMENT<<
The other integration, according to the functionalities criterion, involved two important component of the Power Enjoy system: the Car Controller and the Ride Controller. With the integration of this three internal component, the system can manage a ride.
>>IMMAGINE DEI 3 COMPONENTI INSIEME<<
The third phase involved one of the most imposrtant component of the Power Enjoy System: the List Controller. This particular component manages the employees' list of car, keeping updated which cars need attention. This component is integrateded with the Car Controller: in this way it is guaranteed the maintenance of cars and the employees' management.
>>IMMAGINE DEI DUE COMPONENTI<<
In the fourth phase the Sms Gateway and the Mail Gateway are integrated with the Notification Dispatcher. After the integration of the notification components, the Notification Dispatcher is integrated with the Reservation Controller. This phase guarantees the integrations of the subcomponents involved in the reservation of cars and in the comunications of the neccessary informations to the client.
>>IMMAGINE DEI 4 COMPONENTI INTEGRATI<<
Finally the Reservation Controller is integrated with the List Controller and with the Car Controller, and all of the components mentioned before plus the Account Controller are integrated With the dispatcher.
>>IMMAGINE SOLO DI TUTTI I CONTROLLER E DEL DISPATCHER<<

2.4.2 Subsystem Integration Sequence

>>SEZIONE PER NOI INUTILE IN QUANTO NON ABBIAMO MAI DETTO DI AVERE SOTTOSISTEMI, LA TOGLIEREI E METTEREI LA SITUA FINALE NELLA SEZIONE PRECEDENTE<<

3 Indivual Steps and Test Description

3.1 Account Controller

createUser(allCredentials)

Input	Effect
Some incomplete credentials.	A NullArgumentException is generated.
Unique credentials already exist in the database.	An InvalidArgumentException is generated.
Error in payment information, when they are used in order to withdraw the symbolic payment amount.	An InvalidArgumentException is generated.
Valid input.	The user is inserted in the database. The password is generated randomly in the DBMS and a mail with its is sent to the user.

Table 1: createUser test.

checkLoginCredentials(loginCredentials)

Input	Effect
Some incomplete login credentials.	A NullArgumentException is generated.
Some wrong login credentials.	An InvalidArgumentException is generated.
Valid input.	The system returns the user's personal page.

Table 2: checkLoginCredentials test.

retrieveAccountInformation(user)

Input	Effect
Valid input.	The database returns the user's entry.

Table 3: retrieveAccountInformation test.

modify(user, parameter)

Input	Effect
A null parameter.	A NullArgumentException is generated.
Unique parameter already exist in the database.	An InvalidArgumentException is generated.
Error in payment information, when they are used in order to withdraw the symbolic payment amount.	An InvalidArgumentException is generated.
Valid input.	The database updates the user's entry with the parameter.

Table 4: modify test.

3.2 Car Controller

changePhaseOfCar(carId)

Input	Effect
An invalid carId.	An InvalidArgumentException is generated.
Human error.	An InvalidOperationException is generated.
Valid request.	The database calculate the actual phase, and update the record of the car.

Table 5: changePhaseOfCar test.

<TODO: dire che è il caso in cui l'employee cambia stato ma lo stato non va bene, e la query al database, seguendo le regiole definite in alloy, restituisce uno statoi incompatibile. Riportiamo anche il diagramma di codice di alloy con le definizioni.>

needHelp(stateOfCar)

Input	Effect
No signal for call.	An InvalidOperationException is generated.
Valid request.	The system of the car call the call center.

Table 6: needHelp test.

closeTheCar(carId)

Input	Effect
An invalid carId.	An InvalidArgumentException is generated.
Valid input.	The database calculate the actual phase, and update the record of the car.

Table 7: closeTheCar test.

<TODO: dire che il sistema prende atto della posizione e nel caso manda la multa e l'employee se è unsafe.>

openTheCar(carId)

Input	Effect
An invalid carId.	An InvalidArgumentException is generated.
Valid input.	The system send the correct open message to the car and wait for its response.

Table 8: openTheCar test.

<TODO: verificare Furthermore the database calculate the actual phase, and update the record of the car.However if the response is negative an FailedActionException.>

3.3 List Controller

createTypedListOfCar(listCreator)

Input	Effect
A null parameter.	A NullArgumentException is raised.
Valid input.	The correct List is generated by the system, with a query on the database

Table 9: createTypedListOfCar test.

3.4 Reservation Controller

createReservation(carId)

Input	Effect
A null parameter.	A NullArgumentException is raised.
An invalid carId.	An InvalidArgumentException is generated.
Valid input.	A new entry is added to the reservation database table.

Table 10: createReservation test.

deleteReservation(reservationId)

Input	Effect
A null parameter.	A NullArgumentException is raised.
An invalid reservationId.	An InvalidArgumentException is generated.
Valid input.	The entry corresponding to the reservationId is modified.

Table 11: deleteReservation test.

sendReservationMessage(drivingLicense)

Input	Effect
A null parameter.	A NullArgumentException is raised.
An invalid drivingLicense.	An InvalidArgumentException is generated.
Valid input.	The system use the SMS gateway in order to send the message.

Table 12: sendReservationMessage test.

getRemainigTime(reservationId)

Input	Effect
A null parameter.	A NullArgumentException is raised.
An invalid reservationId.	An InvalidArgumentException is generated.
Valid input.	The system calculate the remaining by using a query on the database with the reservationId.

Table 13: getRemainingTime test.

generatePath(positionOfTheUser, positionOfTheCar)

Input	Effect
A null parameter.	A NullArgumentException is raised.
All parameters are null.	A NullArgumentException is raised.
An invalid reservationId.	An InvalidArgumentException is generated.
All parameters are invalid.	An InvalidArgumentException is generated.
Valid input.	The system sends the request with the two position using Google Map API.

Table 14: generatePath test.

3.5 Ride Controller

finishTheRide(rideId)

Input	Effect
A null rideId.	A NullArgumentException is raised.
An invalid rideId.	An InvalidArgumentException is generated.
Valid input.	The system updates the entry in the database

Table 15: finishTheRide test.

computeFinalPrice(rideId)

Input	Effect
A null rideId.	A NullArgumentException is raised.
An invalid rideId.	An InvalidArgumentException is generated.
Valid input.	The system calculate the final price with a query on the database in order to retrieve the possible discounts.

Table 16: computeFinalPrice test.

payTheRide(rideId)

Input	Effect
A null rideId.	A NullArgumentException is raised.
An invalid rideId.	An InvalidArgumentException is generated.
Valid input.	The system send the correct payment information, taken from the database, to the Payment gateway.

Table 17: payTheRide test.

DISPATCHER

(the dispatcher contains a reference for all the controllers, in order to call the correct method based on the request that are forwarded by that class)

all the method with request before

NOT DISPATCHER

(That dispatcher contains a reference for the notification gateway needed by the system)

SMS EMAIL PARA

send email

send SMS

ALL GATEWAY

with send things

4 Performance Analysis

During the integration test phase there must be present also a performance analysis phase, in order to satisfy the non-functional requirements defined in the RASD document. The main analysis mus be done on the waiting time that the future users of the system, both employees and normal users, have to wait in order to complete their request. In the textbox reported below there are all the requirements mentioned before.

Non-functional requirements

- *When the user finishes his/her ride, he/she has 1 minute to plug the car;*
- *The system closes the car 20 seconds after the user came out from the vehicle;*
- *The system must answer to the user's requests within 10 seconds;*
- *The system must be available 24h/7d.*

The availability of the system must also be tested. The two servers contained in both the server farms, definend in DD document, must be able to keep the system online also if one goes offline due to possible errors or informatic attacks. <TODO:verify> In case of updates of the system, they must be done separately on each server, so one of the test cases has to check the availability of the system during upgrades.
To support the tests for the mobile application, thank to the use of PhoneGap, the application can be tested only on one operating mobile system. In actual fact the same scenarios of tests can be executed on different platforms. In section 5 there are the software that can be used in order to test an application builded with PhoneGap.
In the RASD document are mentioned the minimum software requirements that the mobile phone has to provide in order to run correctly the PowerEnjoy application. That requirements are mapped into harware requirements for each mobile OS, so the application must be done in order to be executed with resonable time based on that hardware specification. In the textbox there is a quick view of what we have just mentioned.

To use the mobile application:

- *A smartphone with:*
  - *Android: KitKat (4.4) and later;*
  - *iOS: version 6 or later;*
  - *Windows Mobile: version 10 or later;*
- *An active number, with a plans that contains internet connection:*
  - *3G;*
  - *4G: not required, but recommended.*
- *An amount of space in order to install the application.*

To view the harware specification follow the link reported in table.

OS	Link to harware requirements
Windows 10 Mobile	https://msdn.microsoft.com/en-us/windows/hardware/commercialize/design/minimum/minimum-hardware-requirements-overview
Android KitKat	https://static.googleusercontent.com/media/source.android.com/it//compatibility/4.4/android-4.4-cdd.pdf
IOS 6	see the specification of the Iphone 4

The dimension of the final package of the application must theoretically not exceed 50 MB. Based on that specification, and on what is mentioned in this section, the final mobile application must not exceed 100 MB of RAM on every devices and the medium value of RAM occupied by the application has to be about 80 MB.

5 Tools and Test Equipment Required

PhoneGap http://phonegap.com/blog/2015/10/27/testmunk-guest-post/

6 Program Stubs and Test Data Required

6.1 Program Stubs and Test Data

7 Appendices

7.1 Used Tools

- Microsoft Visio 2016: for all the diagrams in that document (as Testing Diagrams, etc ...);
- Lyx document processor: to write all the document;
- SourceTree: used as GitHub manager;
- GitHub: used to manage the shared building process of that document.

8 Hours of Work

Day	Guido Muscioni	Marco Orbelli	Paola Marchesini
2/01	1		
4/01	2		
5/01			
6/01			
7/01			
8/01	4	4	4
9/01		2	
10/01			
11/01	5		5
12/01			
13/01			
14/01			
15/01			
total			

List of Figures