

**PowerEnJoy Project**



**Integration Test Plan Document**

**(ITPD)**

Alfredo Maria Fomitchenko (mat. 874656)

Version: 1.0

Release date: 15 January 2017

1. Introduction 4

1.1 Revision History 4

1.2 Purpose and Scope 4

1.3 List of Definitions and Abbreviations 5

1.4 List of Reference Documents  5

2. Integration Strategy 6

2.1 Entry Criteria 6

2.2 Elements to be Integrated 6

2.3 Integration Testing Strategy 7

2.4 Sequence of Components/Function Integration 7

2.4.1 Critical-module-first Integration Testing: Router 8

2.4.2 Critical-module-first Integration Testing: DataController 10

2.4.3 Bottom-up Integration Testing: ListAvailableCarsHandler 12

2.4.4 Bottom-up Integration Testing: RegistrationManager 13

2.4.5 Bottom-up Integration Testing: ChargeManager 14

2.4.6 Bottom-up Integration Testing: Front End 15

3. Individual Steps and Test Description 16

3.1 Integration test case I1 16

3.2 Integration test case I2 17

3.3 Integration test case I3 18

3.4 Integration test case I4 19

3.5 Integration test case I5 23

3.6 Integration test case I6 27

3.7 Integration test case I7 29

3.8 Integration test case I8 33

3.9 Integration test case I9 34

3.10 Integration test case I10 35

3.11 Integration test case I11 37

3.12 Integration test case I12 40

3.13 Integration test case I13 43

3.14 Integration test case I14 46

3.15 Integration test case I15 49

3.16 Integration test case I16 50

3.17 Integration test case I18 54

3.18 Integration test case I19 57

3.19 Integration test case I20 59

3.20 Integration test case I21 60

3.21 Integration test case I22 61

3.22 Integration test case I23 63

4. Tools and Test Equipment Required 69

4.1 Tools 69

4.2 Test Equipment 70

5. Program Stubs and Test Data Required 71

6. Effort 74

# Introduction

## Revision History

* v1.0

## Purpose and Scope

This project aims to provide customers within the administrative division of Milan with a car sharing service and all the associated functionalities as widely discussed in the RASD and the DD.

This document represents the Integration Test Plan Document, which outlines the organization of the integration testing activities aiming to make the different components of the system correctly interoperate and avoid any unexpected behavior.

## List of Definitions and Abbreviations

In extension to the RASD and DD Definitions, acronyms, abbreviations paragraphs, below are some definitions and abbreviations used in this document:

* “J2EE” = abbreviation for Java Enterprise Edition, platform-independent Java-centric environment considered within the scope of this project as the fundamental system runtime context.
* “IDE” = acronym for Integrated Development Environment, software application that provides the comprehensive facilities for software development.

## List of Reference Documents

I used for this DD as reference for the project assignment, the general layout and structure

* Assignments AA 2016-2017.pdf,
* RASD v1.1 Alfredo Fomitchenko.pdf,
* DD v1.1 Alfredo Fomitchenko.pdf,
* Integration Test Plan Example.pdf (SPINGRID example),
* Integration testing example document.pdf (MyTaxiService example);

to create the UML diagrams

* draw.io website;

to write the actual document

* Microsoft Word 2016;

as scheduling and time effort management tracker

* Microsoft Excel 2016.

# Integration Strategy

## Entry Criteria

Integration testing can produce meaningful results only if some conditions about the overall project progress are met.

The first fundamental criteria is that the Requirements Analysis and Specifications Document and the Design Document must be released in a fully revised way: this assures that the architecture and the interactions concerning the components has been settled and agreed upon, which gives the definitive picture of what should interact and work with what component.

The second criteria, as important as the first, is that every component that has been developed individually must be tested within its own scope, assuring that problems raised in the integration phase are not due to internal algorithmic issues.

## Elements to be Integrated

As extensively illustrated in the Design Document, the system will be composed of low-level components and external gateways, which wholly constitute and provide the necessary functionalities. These are: AuthenticationManager, Router, ListAvailableCarsHandler, LoginManager, RegistrationManager, ReservationManager, RideManager, LockManager, TimerManager, ChargeManager, DiscountHelper, DataController, Cache, Database Server, LocatorHelper, GoogleMapsGateway, EmailServiceProviderGateway, BankGateway, Customer application, Car service.

For all the details about the integration testing, see the following paragraphs.

## Integration Testing Strategy

The Design Document highlights two critical components:

1. Router, as every incoming request from the mobile application and every request from one component to another must be correctly interpreted and forwarded;
2. DataController, as the world model and logic regarding customers, cars, reservations and rides lies within it and every other component trying to read, write or manipulate data asks this one to do so.

The importance of these two components and their functions naturally leads to the choice of starting the whole integration testing process with a focus onto the two given by a critical-module-first approach.

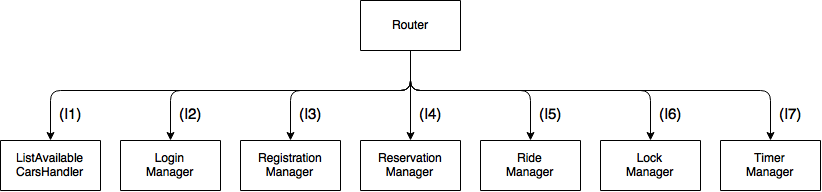
Then, given Router and DataController are fully integrated with the components they communicate directly with, a hierarchical bottom-up approach follows, for which low-level components are found developed and ready to be integrated.

## Sequence of Components/Function Integration

As mentioned, the first integration phase focuses onto Router and DataController which represent critical parts of the system. Then, a more hierarchical bottom-up approach follows.

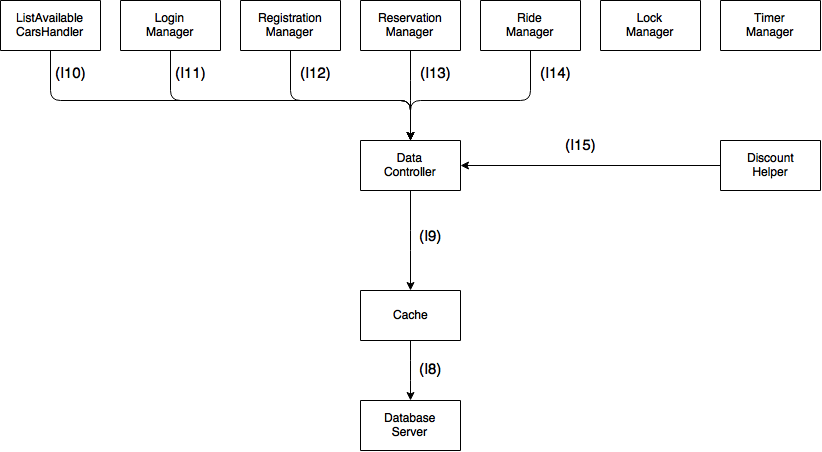
Please note that, especially regarding Router testing, diagram arrows going from A to B mean that B must have already been developed for the whole test to start, but methods are also called the other round (for further details see Individual Steps and Test Description section).

### Critical-module-first Integration Testing: Router



|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I1 | Router → ListAvailableCarsHandler | 3.1 |
| I2 | Router → LoginManager | 3.2 |
| I3 | Router → RegistrationManager | 3.3 |
| I4 | Router → ReservationManager | 3.4 |
| I5 | Router → RideManager | 3.5 |
| I6 | Router → LockManager | 3.6 |
| I7 | Router → TimerManager | 3.7 |

### Critical-module-first Integration Testing: DataController



|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I8 | Cache → Database Server | 3.8 |
| I9 | DataController → Cache | 3.9 |
| I10 | ListAvailableCarsHandler → DataController | 3.10 |
| I11 | LoginManager → DataController | 3.11 |
| I12 | RegistrationManager → DataController | 3.12 |
| I13 | ReservationManager → DataController | 3.13 |
| I14 | RideManager → DataController | 3.14 |
| I15 | DiscountHelper → DataController | 3.15 |

### Bottom-up Integration Testing: ListAvailableCarsHandler

|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I16 | LocatorHelper → GoogleMapsGateway | 3.16 |
| I17 | ListAvailableCarsHandler→ LocatorHelper | 3.17 |



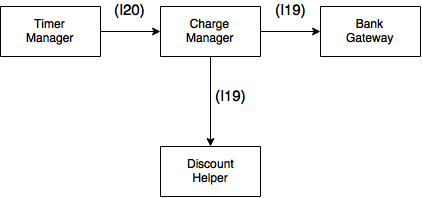
### Bottom-up Integration Testing: RegistrationManager

|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I18 | RegistrationManager → EmailServiceProviderGateway | 3.18 |

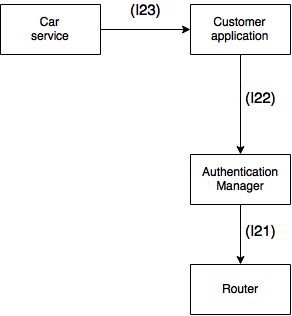


### Bottom-up Integration Testing: ChargeManager

|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I19 | ChargeManager → DiscountHelper, BankGateway | 3.19 |
| I20 | TimerManager → ChargeManager | 3.20 |



### Bottom-up Integration Testing: Front End



|  |  |  |
| --- | --- | --- |
| ID | Integration Test | Description paragraph |
| I21 | AuthenticationManager → Router | 3.21 |
| I22 | Customer application → AuthenticationManager | 3.22 |
| I23 | Car service → Customer application | 3.23 |

# Individual Steps and Test Description

## Integration test case I1

|  |  |
| --- | --- |
| **Test case**  **identifier** | I1T1 |
| **Integration**  **test items** | Router → ListAvailableCarsHandler |
| **Input**  **specification** | Create the Router request triggered by the customer to retrieve the available cars from the customer’s position given as either an address or a GPS position |
| **Output specification** | Check if the correct ListAvailableCarsHandler methods are called |
| **Environmental needs** | AuthenticationManager driver |

## Integration test case I2

|  |  |
| --- | --- |
| **Test case**  **identifier** | I2T1 |
| **Integration**  **test items** | Router → LoginManager |
| **Input**  **specification** | Create the Router request triggered by the customer to log into the system |
| **Output specification** | Check if the correct LoginManager methods are called |
| **Environmental needs** | AuthenticationManager driver |

## Integration test case I3

|  |  |
| --- | --- |
| **Test case**  **identifier** | I3T1 |
| **Integration**  **test items** | Router → RegistrationManager |
| **Input**  **specification** | Create the Router request triggered by the customer to register a new account for logging into the system |
| **Output specification** | Check if the correct RegistrationManager methods are called |
| **Environmental needs** | AuthenticationManager driver |

## Integration test case I4

|  |  |
| --- | --- |
| **Test case**  **identifier** | I4T1 |
| **Integration**  **test items** | Router → ReservationManager |
| **Input**  **specification** | Create the Router request triggered by the customer to check if he is able to reserve a car given that he has not previously reserved another one |
| **Output specification** | Check if the correct ReservationManager methods are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I4T2 |
| **Integration**  **test items** | ReservationManager → Router |
| **Input**  **specification** | Create the ReservationManager request intended for TimerManager  to start a timer associated to a new reservation |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I4T3 |
| **Integration**  **test items** | Router → ReservationManager |
| **Input**  **specification** | Create the Router request triggered by LockManager to check whether the customer requesting a car unlock is the one that reserved the car |
| **Output specification** | Check if the correct ReservationManager methods are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I4T4 |
| **Integration**  **test items** | ReservationManager → Router |
| **Input**  **specification** | Create the ReservationManager request intended for TimerManager  to stop a timer associated to a reservation |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

## Integration test case I5

|  |  |
| --- | --- |
| **Test case**  **identifier** | I5T1 |
| **Integration**  **test items** | Router → RideManager |
| **Input**  **specification** | Create the Router request triggered by the customer to start a ride |
| **Output specification** | Check if the correct RideManager methods are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I5T2 |
| **Integration**  **test items** | RideManager → Router |
| **Input**  **specification** | Create the RideManager request  intended for TimerManager  to start a timer associated to a new ride |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I5T3 |
| **Integration**  **test items** | RideManager → Router |
| **Input**  **specification** | Create the RideManager request  intended for TimerManager  to stop a timer associated to a ride |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I5T4 |
| **Integration**  **test items** | RideManager → Router |
| **Input**  **specification** | Create the RideManager request  intended for LockManager  to lock the car |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

## Integration test case I6

|  |  |
| --- | --- |
| **Test case**  **identifier** | I6T1 |
| **Integration**  **test items** | LockManager → Router |
| **Input**  **specification** | Create the LockManager request  intended for ReservationManager  to check whether the customer requesting a car unlock is the one that reserved the car |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I6T2 |
| **Integration**  **test items** | LockManager → Router |
| **Input**  **specification** | Create the LockManager request  intended for the car service  to lock the car |
| **Output specification** | Check if the correct Router methods  are called |
| **Environmental needs** | None |

## Integration test case I7

|  |  |
| --- | --- |
| **Test case**  **identifier** | I7T1 |
| **Integration**  **test items** | Router → TimerManager |
| **Input**  **specification** | Create the Router request triggered by ReservationManager to start a reservation timer |
| **Output specification** | Check if the correct TimerManager methods  are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I7T2 |
| **Integration**  **test items** | Router → TimerManager |
| **Input**  **specification** | Create the Router request triggered by ReservationManager to stop a reservation timer |
| **Output specification** | Check if the correct TimerManager methods  are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I7T3 |
| **Integration**  **test items** | Router → TimerManager |
| **Input**  **specification** | Create the Router request triggered by RideManager to start a ride timer |
| **Output specification** | Check if the correct TimerManager methods  are called |
| **Environmental needs** | AuthenticationManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I7T4 |
| **Integration**  **test items** | Router → TimerManager |
| **Input**  **specification** | Create the Router request triggered by RideManager to stop a ride timer |
| **Output specification** | Check if the correct TimerManager methods  are called |
| **Environmental needs** | AuthenticationManager driver |

## Integration test case I8

|  |  |
| --- | --- |
| **Test case**  **identifier** | I8T1 |
| **Integration**  **test items** | Cache → Database Server |
| **Input**  **specification** | Create the Cache request triggered by DataController to retrieve the appropriate data necessary for DataController to fulfill the call |
| **Output specification** | Check if Database Server correctly handles the call |
| **Environmental needs** | Cache and Database Server are intended to be fully integrated as parts of an Oracle DBMS component of the Red Hat OpenShift cloud infrastructure |

## Integration test case I9

|  |  |
| --- | --- |
| **Test case**  **identifier** | I9T1 |
| **Integration**  **test items** | DataController → Cache |
| **Input**  **specification** | Create all DataController requests intended for Database Server to retrieve the appropriate data |
| **Output specification** | Check if Cache correctly handles the calls |
| **Environmental needs** | Cache and Database Server are intended to be fully integrated as parts of an Oracle DBMS component of the Red Hat OpenShift cloud infrastructure |

## Integration test case I10

|  |  |
| --- | --- |
| **Test case**  **identifier** | I10T1 |
| **Integration**  **test items** | ListAvailableCarsHandler → DataController |
| **Input**  **specification** | Create the ListAvailableCarsHandler request to extract the available cars given the customer’s position |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I10T2 |
| **Integration**  **test items** | ListAvailableCarsHandler → DataController |
| **Input**  **specification** | Create the ListAvailableCarsHandler request to extract the available cars passing as arguments a not correctly formatted latitude and longitude pair |
| **Output specification** | Check if DataController correctly returns a Null cars list |
| **Environmental needs** | Cache stub |

## Integration test case I11

|  |  |
| --- | --- |
| **Test case**  **identifier** | I11T1 |
| **Integration**  **test items** | LoginManager → DataController |
| **Input**  **specification** | Create the LoginManager request to retrieve the password associated to the customer email address |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I11T2 |
| **Integration**  **test items** | LoginManager → DataController |
| **Input**  **specification** | Create the LoginManager request to retrieve the password associated to the customer email address providing an email address no database record corresponds to |
| **Output specification** | Check if DataController correctly returns a Null *pwd* string variable |
| **Environmental needs** | Cache stub |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I11T3 |
| **Integration**  **test items** | LoginManager → DataController |
| **Input**  **specification** | Create the LoginManager request to cache the customer data |
| **Output specification** | Check if the correct DataController methods  are called |
| **Environmental needs** | Router driver |

## Integration test case I12

|  |  |
| --- | --- |
| **Test case**  **identifier** | I12T1 |
| **Integration**  **test items** | RegistrationManager → DataController |
| **Input**  **specification** | Create the RegistrationManager request to check whether the customer has already registered to the system |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I12T2 |
| **Integration**  **test items** | RegistrationManager → DataController |
| **Input**  **specification** | Create the RegistrationManager request to check whether the customer has already registered to the system passing as argument a customer who has already registered |
| **Output specification** | Check if DataController correctly throws an error |
| **Environmental needs** | Cache stub |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I12T3 |
| **Integration**  **test items** | RegistrationManager → DataController |
| **Input**  **specification** | Create the RegistrationManager request to generate a new customer’s record into the database |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

## Integration test case I13

|  |  |
| --- | --- |
| **Test case**  **identifier** | I13T1 |
| **Integration**  **test items** | ReservationManager → DataController |
| **Input**  **specification** | Create the ReservationManager request to check whether the customer either has already reserved a car or is currently riding a car |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |
| **Comments** | ReservationManager unit testing checks whether the component correctly throws an error when either a ride or a reservation associated to the customer has been found |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I13T2 |
| **Integration**  **test items** | ReservationManager → DataController |
| **Input**  **specification** | Create the ReservationManager request to generate a new reservation associating customer and car |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I13T3 |
| **Integration**  **test items** | ReservationManager → DataController |
| **Input**  **specification** | Create the ReservationManager request to generate a new reservation associating customer and car when the car is not available anymore |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |
| **Comments** | The input specification occurs when the customer had correctly retrieve the car when it was available, but meanwhile he was deciding to reserve it another customer reserved it |

## Integration test case I14

|  |  |
| --- | --- |
| **Test case**  **identifier** | I14T1 |
| **Integration**  **test items** | RideManager → DataController |
| **Input**  **specification** | Create the RideManager request to generate a new ride associating customer and car |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I14T2 |
| **Integration**  **test items** | RideManager → DataController |
| **Input**  **specification** | Create the RideManager request to insert among the ride information the battery percentage charge |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I14T3 |
| **Integration**  **test items** | RideManager → DataController |
| **Input**  **specification** | Create the RideManager request to set a car available again after a ride has been terminated |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | None |

## Integration test case I15

|  |  |
| --- | --- |
| **Test case**  **identifier** | I15T1 |
| **Integration**  **test items** | DiscountHelper → DataController |
| **Input**  **specification** | Create the DiscountHelper request to retrieve the information associated to the terminated ride |
| **Output specification** | Check if the correct DataController methods are called |
| **Environmental needs** | ChargeManager driver |

## Integration test case I16

|  |  |
| --- | --- |
| **Test case**  **identifier** | I16T1 |
| **Integration**  **test items** | LocatorHelper → GoogleMapsGateway |
| **Input**  **specification** | Create the LocatorHelper request to convert an address into a pair of latitude and longitude coordinates via GeoCoding API |
| **Output specification** | Check if GoogleMapsGateway correctly handles the call |
| **Environmental needs** | ListAvailableCarsHandler driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I16T2 |
| **Integration**  **test items** | LocatorHelper → GoogleMapsGateway |
| **Input**  **specification** | Create the LocatorHelper request to retrieve the customer’s position via GeoLocation API |
| **Output specification** | Check if GoogleMapsGateway correctly handles the call |
| **Environmental needs** | ListAvailableCarsHandler driver |

Integration test case I17

|  |  |
| --- | --- |
| **Test case**  **identifier** | I17T1 |
| **Integration**  **test items** | ListAvailableCarsHandler → LocatorHelper |
| **Input**  **specification** | Create the ListAvailableCarsHandler request to convert an address into a pair of latitude and longitude coordinates via GeoCoding API |
| **Output specification** | Check if the correct LocatorHelper methods are called |
| **Environmental needs** | Router driver |
| **Comments** | LocatorHelper unit testing checks whether the component returns a false *found* variable if a not valid address has been provided |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I17T2 |
| **Integration**  **test items** | ListAvailableCarsHandler → LocatorHelper |
| **Input**  **specification** | Create the ListAvailableCarsHandler request to retrieve the customer’s position via GeoLocation API |
| **Output specification** | Check if the correct LocatorHelper methods are called |
| **Environmental needs** | Router driver |

## Integration test case I18

|  |  |
| --- | --- |
| **Test case**  **identifier** | I18T1 |
| **Integration**  **test items** | RegistrationManager → EmailServiceProviderGateway |
| **Input**  **specification** | Create the RegistrationManager request to check whether the email address provided by the customer is a valid one |
| **Output specification** | Check if EmailServiceProviderGateway correctly handles the call |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I18T2 |
| **Integration**  **test items** | RegistrationManager → EmailServiceProviderGateway |
| **Input**  **specification** | Create the RegistrationManager request to check whether the email address provided by the customer is a valid one passing as argument a not valid email address |
| **Output specification** | Check if EmailServiceProviderGateway correctly returns an error |
| **Environmental needs** | Router driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I18T3 |
| **Integration**  **test items** | RegistrationManager → EmailServiceProviderGateway |
| **Input**  **specification** | Create the RegistrationManager request to send the email containing the customer’s associated password |
| **Output specification** | Check if EmailServiceProviderGateway correctly handles the call |
| **Environmental needs** | Router driver |

## Integration test case I19

|  |  |
| --- | --- |
| **Test case**  **Identifier** | I19T1 |
| **Integration**  **test items** | ChargeManager → DiscountHelper, BankGateway |
| **Input**  **Specification** | Create the ChargeManager request to retrieve the discount applicable to the customer’s ride charge thanks to his virtuous behaviors |
| **Output specification** | Check if the correct DiscountHelper methods are called |
| **Environmental needs** | TimerManager driver |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I19T2 |
| **Integration**  **test items** | ChargeManager → DiscountHelper, BankGateway |
| **Input**  **specification** | Create the ChargeManager request to charge the customer of the ride charge amount after any applicable discount |
| **Output specification** | Check if BankGateway correctly handles the call |
| **Environmental needs** | TimerManager driver, Cache stub |

## Integration test case I20

|  |  |
| --- | --- |
| **Test case**  **Identifier** | I20T1 |
| **Integration**  **test items** | TimerManager → ChargeManager |
| **Input**  **Specification** | Create the TimerManager request to charge the customer after an associated ride timer has been stopped |
| **Output specification** | Check if the correct ChargeManager methods are called |
| **Environmental needs** | I19 succeeded |

## Integration test case I21

|  |  |
| --- | --- |
| **Test case**  **Identifier** | I21T1 |
| **Integration**  **test items** | AuthenticationManager → Router |
| **Input**  **Specification** | Create the AuthenticationManager validated forwarding request to Router |
| **Output specification** | Check if the correct Router methods are called |
| **Environmental needs** | Customer application driver |

## Integration test case I22

|  |  |
| --- | --- |
| **Test case**  **Identifier** | I22T1 |
| **Integration**  **test items** | Customer application → AuthenticationManager |
| **Input**  **Specification** | Create any Customer application request shown in the Design Document sequence diagrams |
| **Output specification** | Check if AuthenticationManager validates correctly the calls |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I22T2 |
| **Integration**  **test items** | Customer application → AuthenticationManager |
| **Input**  **specification** | Create fraudulent customer requests sent by unofficial mobile applications or mobile application emulators |
| **Output specification** | Check if AuthenticationManager does not correctly authorize the communication to Router |
| **Environmental needs** | None |
| **Comments** | The typical scenarios which this aims to test are calls emulating a car request to be unlocked performed by a customer which is not the legit customer that reserved the car |

## Integration test case I23

|  |  |
| --- | --- |
| **Test case**  **Identifier** | I23T1 |
| **Integration**  **test items** | Customer application → Car service |
| **Input**  **Specification** | Create the Customer application request to establish a Bluetooth connection |
| **Output specification** | Check if Car service correctly activates the necessary Bluetooth interface capabilities |
| **Environmental needs** | I22 succeeded |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I23T2 |
| **Integration**  **test items** | Customer application → Car service |
| **Input**  **specification** | Create the Customer application request to unlock the car |
| **Output specification** | Check if the correct Car service methods are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I23T3 |
| **Integration**  **test items** | Car service → Customer application |
| **Input**  **specification** | Create the Car service request to check whether the car can be unlocked by the customer |
| **Output specification** | Check if the correct Customer application methods are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I23T4 |
| **Integration**  **test items** | Car service → Customer application |
| **Input**  **specification** | Create the Car service request to provide to the system the current ride passengers number |
| **Output specification** | Check if the correct Customer application methods are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I23T5 |
| **Integration**  **test items** | Car service → Customer application |
| **Input**  **specification** | Create the Car service request to provide to the system the current battery charge at the end of a ride |
| **Output specification** | Check if the correct Customer application methods are called |
| **Environmental needs** | None |

|  |  |
| --- | --- |
| **Test case**  **identifier** | I23T6 |
| **Integration**  **test items** | Customer application → Car service |
| **Input**  **specification** | Create the Customer application request to perform an automated lock after the customer closes the driver door |
| **Output specification** | Check if the correct Car service methods are called |
| **Environmental needs** | None |

# Tools and Test Equipment Required

## Tools

Alongside with manual operations and test data planning and writing, testing activities on the Java Enterprise Edition environment will take advantage of automated tools.

The JUnit framework will clearly take an important part of the unit testing activities for each component, but can also be involved in integration checks regarding for example objects returned by methods, errors thrown among components, etc.

However, Arquillian will be the main tool for in-container integration testing. The Arquillian framework allows the execution of the test cases extensively illustrated in this document against the Java runtime container, checking whether the interaction between a component and the surrounding environment works correctly.

Also, when dependency injection and the interaction with the database server are concerned, Arquillian represents a very powerful and widely used tool.

A significant portion of the testing activities will also focus onto the Android mobile application to be developed as fundamental part for providing the car sharing service to the customers.

Android Studio, the official IDE for Android app development and testing, industry standard officially supported by Google itself, will be the main support tool for the Android environment. Moreover, within and in extension to the IDE, the widely testing features involved in the process will be UI Automator (to build UI tests simulating user touchscreen interactions) and AndroidJUnitRunner (a class used to specifically run JUnit tests against an Android package).

## Test Equipment

The cloud infrastructure chosen for the application final deployment is the Red Hat OpenShift one. Clearly the test environment where components will be tested has to be as close to the mentioned as possible, for example a scaled down version of the mentioned infrastructure.

For what concerns the mobile application testing, Android Studio provides a very reliable tool called Android Emulator which simulates different devices onto which prototype, develop and test applications. However, physical devices will be required to deliver a comprehensive testing activity, so at least one Android smartphone for each display size from three to six inches at steps of half an inch will be used.

# Program Stubs and Test Data Required

Stubs needed will be

* Cache stub, that will simulate the response for database data requests coming from DataController, and
* DataController stub, that in particular will play a fundamental role within the integration testing of the components involved in the customer charge process (see I19).

Moreover, following are the drivers to be developed, which will invoke methods on the components to perform the integrations indicated below:

|  |  |
| --- | --- |
| Driver | to perform integration testing between |
| AuthenticationManager  driver | Router → ListAvailableCarsHandler  Router → RegistrationManager  Router → LoginManager  Router → ReservationManager  Router → RideManager  Router → LockManager  Router → TimerManager |
| Router  driver | ListAvailableCarsHandler → DataController  RegistrationManager → DataController  LoginManager → DataController  ReservationManager → DataController  RideManager → DataController |
| ChargeManager  driver | DiscountHelper → DataController |
| ListAvailableCarsHandler  driver | LocatorHelper → GoogleMapsGateway |
| TimerManager  driver | ChargeManager → BankGateway |
| Customer application  driver | AuthenticationManager → Router |

For what concerns about the test data, every test case input specification provides the information necessary to the planning and writing of it.

In particular, test case I22T2 concerning the development of unofficial mobile applications or mobile application emulators to simulate fraudulent customer requests (for example to ask for a car unlock) will definitely require more effort and the expertise of security specialists.

# Effort

* 18 December 2016: **2,2 h**
* 20 December 2016: **1 h**
* 23 December 2016: **1 h**
* 26 December 2016: **2,5 h**
* 27 December 2016: **2,6 h**
* 28 December 2016: **3,4 h**
* 29 December 2016: **3 h**
* 5 January 2016: **1,3 h**
* 6 January 2016: **0,5 h**
* 7 January 2016: **3,3 h**
* 8 January 2016: **1 h**
* 10 January 2016: **1 h**
* 12 January 2016: **1,1 h**

**23,9 h**