POLITECNICO
MILANO 1863

Software Engineering 2 A.A. 2015-16

Integration Test Plan Document

for

MyTaxiService

by Francesco Picciotti (854021)

and Marco Travaglini (859186)

Table of Contents

1. Introduction

1.1 Revision History

Record all revisions to the document.

Date	Reason For Changes	Version
21/01/2016	First submission of the document	1.0

1.2 Purpose and Scope

1.2.1 Purpose

The following document describes how the testing must be done during the software's development. Since this part is a tricky phase, it should follow a rigorous approach which basically is based on the architectural decisions done during the design phase and the goals aimed during the requirement analysis, that respectively are written in the Design Document and the Requirement Analysis Specification Document. This document, indeed, explains how the necessary tests has to be done in order to ensure the interaction between components of the systems, on the other hand it presumes that all the components works properly which means each component has to be tested alone before in order to verify the right working.

1.2.2 Scope

The software, called MyTaxiService, allows regulating the taxi service offered in a city, speeding up the process of finding a taxi, through a mobile app or web application. In fact, a normal user can register himself as a customer or as a taxi driver. A customer can look for a ride stating his current location or a reservation adding the time in which the taxi has to pick him up. A taxi driver, on the other hand, can show his availability to the application and necessarily allow the system to retrieve his current position from the gps localizer and then to reply to a ride or reservation request.

1.3 List of Definitions and Abbreviations

- *QA: Quality Assurance, the team that is concerned in software's verification.*
- *BE: Back End.*

1.4 List of Reference Documents

- *Assignment 4: integration test plan.pdf*
- *Requirements Analysis Specification Document: MyTaxiService AA 2015/16.pdf*
- *Design Document: MyTaxiService AA 2015/16.pdf*
- *Integration Test Plan: SpinGrid.pdf*

2. Integration Strategy

2.1 Entry Criteria

- *All test hardware platforms must have been successfully installed, configured and functioning properly.*
- *All standard software tools including testing tools must have been successfully installed and functioning properly.*
- *All documentation and design of the architecture must be completed and made available.*
- *All people involved in the system test effort must be trained in tools to be used during testing process.*
- *A separate QA environment (with its own webserver, database and Application server instance) must be available.*
- *All the classes belonging to each component must be properly tested.*

2.2 Elements to be Integrated

These listed below are components to be integrated, which the system is made of.

- *CustomerBE:*
  1. *RequestForwarder*
  2. *PastReqEngine*

- *Accounting Manager*

- *TaxiDriverDB*

- *DriverBE:*
  1. *NotificationEngine*
  2. *RequestEngine*
  3. *RequestDisplayer*
  4. *GpsLocalizer*
  5. *AvailableSetter*

- *MyTaxiServiceBE:*
  1. *RequestHandler*
  2. *QueueEngine*
  3. *NotificationDispatcher*

- *DataStorage:*
  1. *RequestRegister*
  2. *AccountingWarehouse*

2.3 Integration Testing Strategy

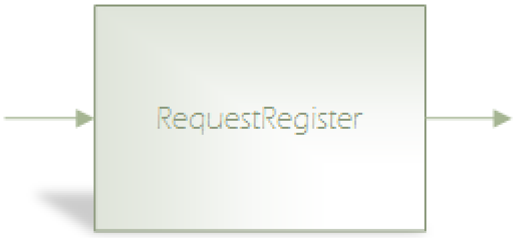
For integration testing of the system, we decide to use the bottom-up approach. First, it requires unit testing, so each component at lower hierarchy are tested individually; then the progressively higher-level combinations of components are tested up to the highest level in order to ensure even a correct interaction among components that have a dependency each other. This strategy allow developing and testing at the same time, so the product will build-up faster since it will be gradually verified in compliance with the project specifications, which are given with the related RASD and DD.

2.4 Sequence of Component/Function Integration

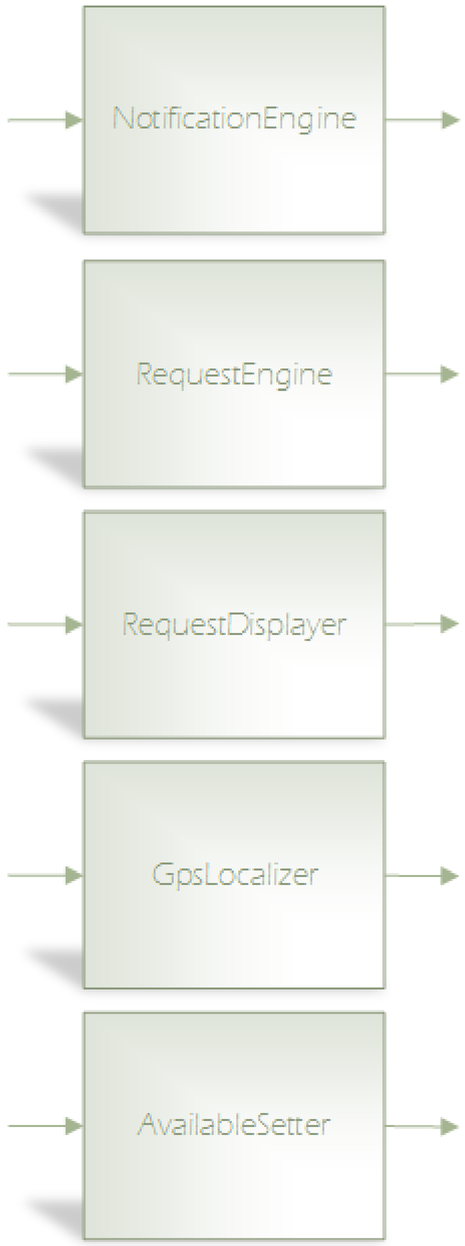
2.4.1 Software Integration Sequence.

- ***CustomerBE***

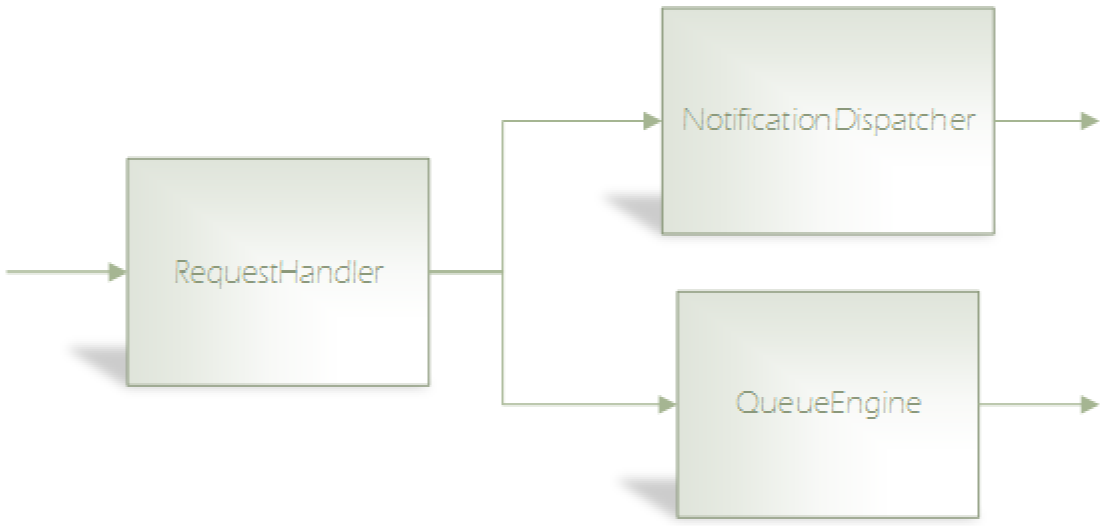
- ***DataStorage***



- ***DriverBE***

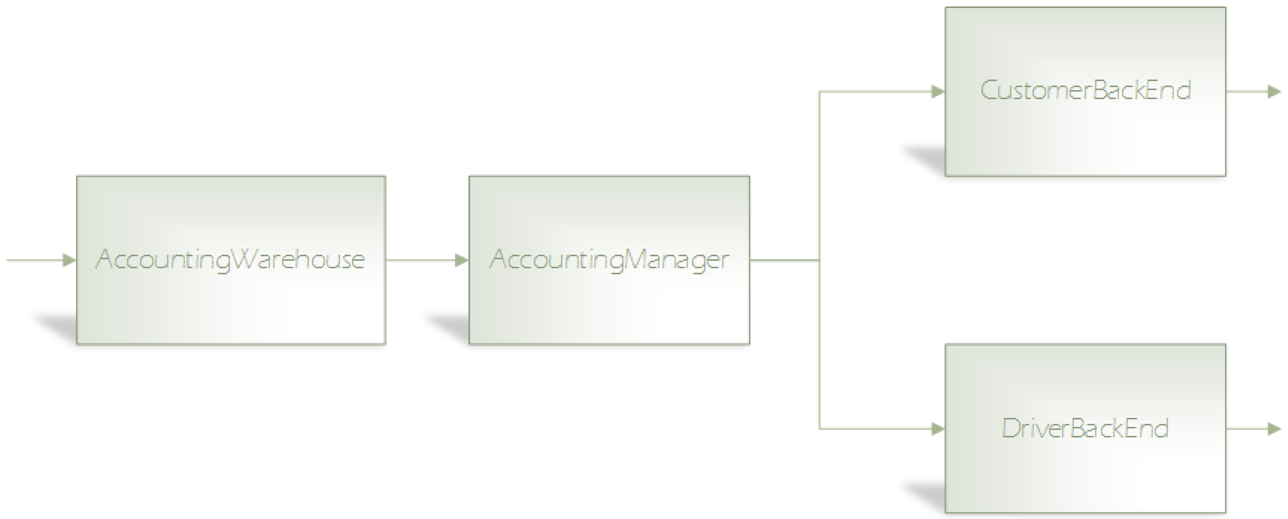

NotificationEngine
RequestEngine
RequestDisplayer
GpsLocalizer
AvailableSetter

❖ ***MyTaxiServiceBE***

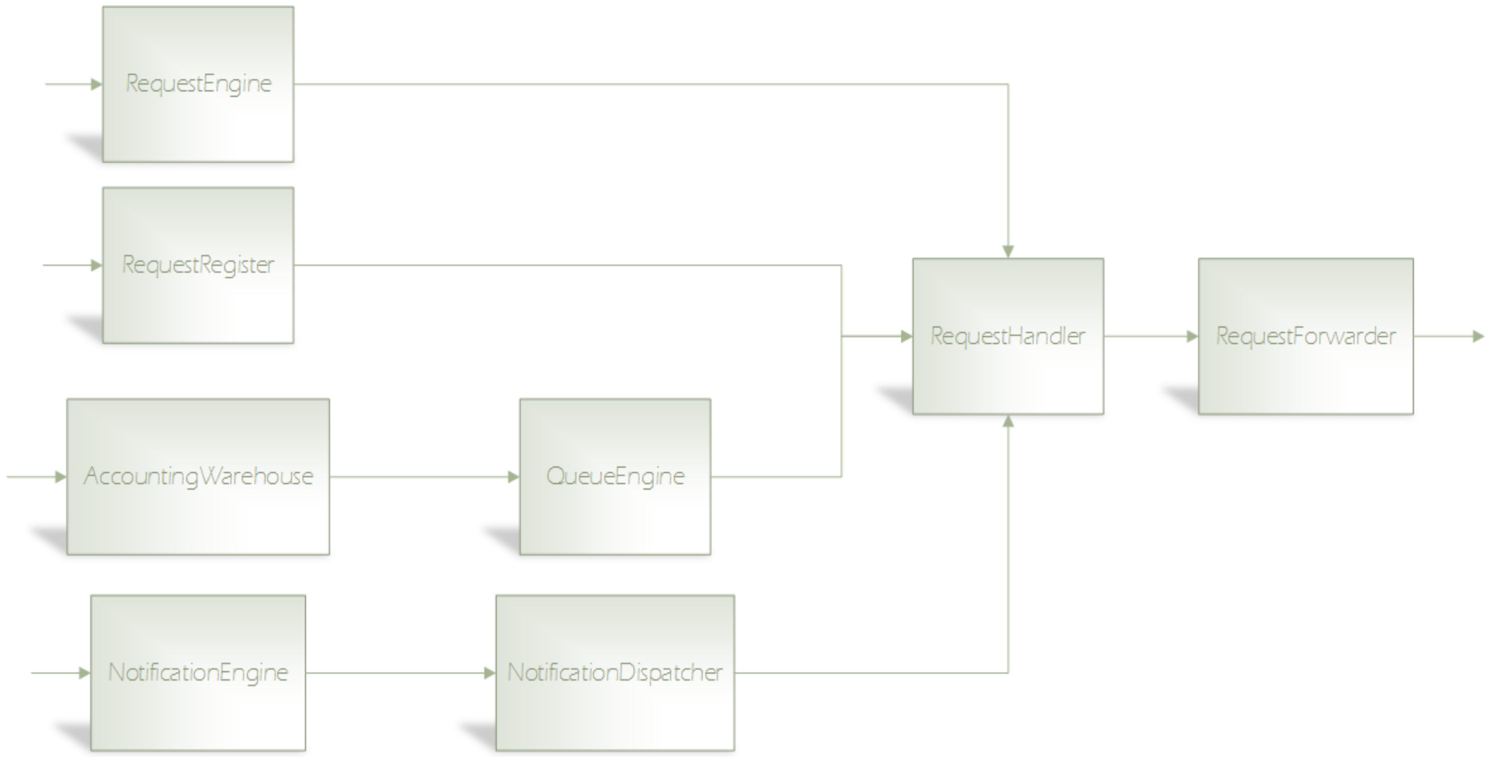


For an explanatory matters, there are shown above also the components' integration belonging to different subsystems, divided by functionalities to be tested.

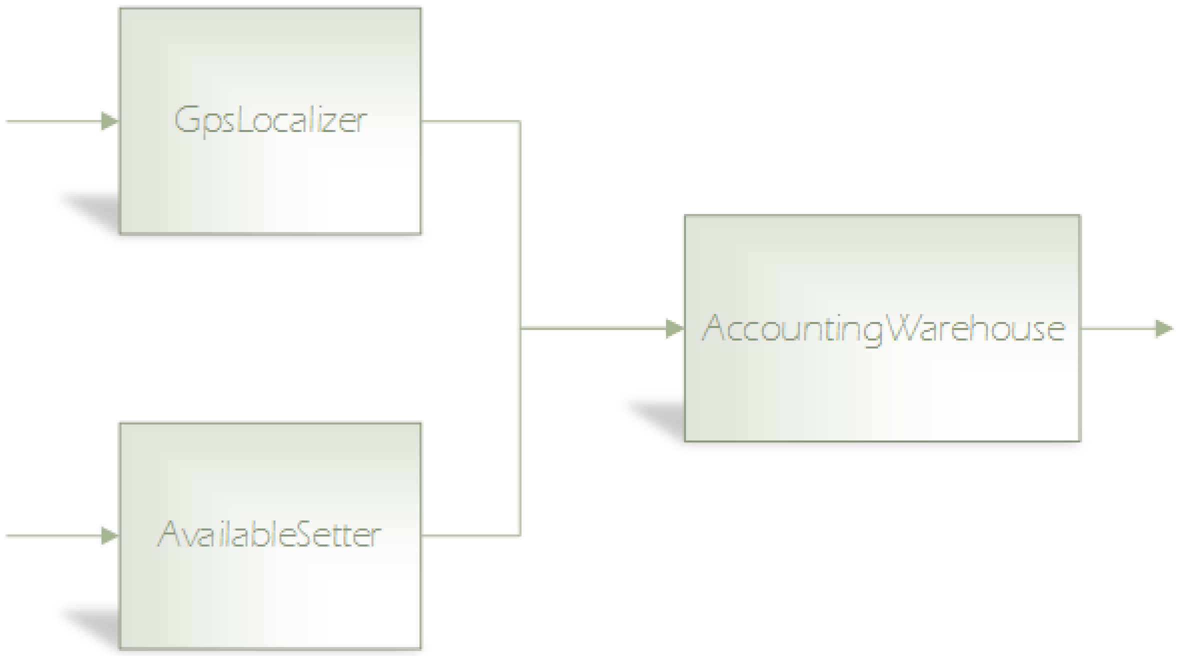
❖ ***Login***



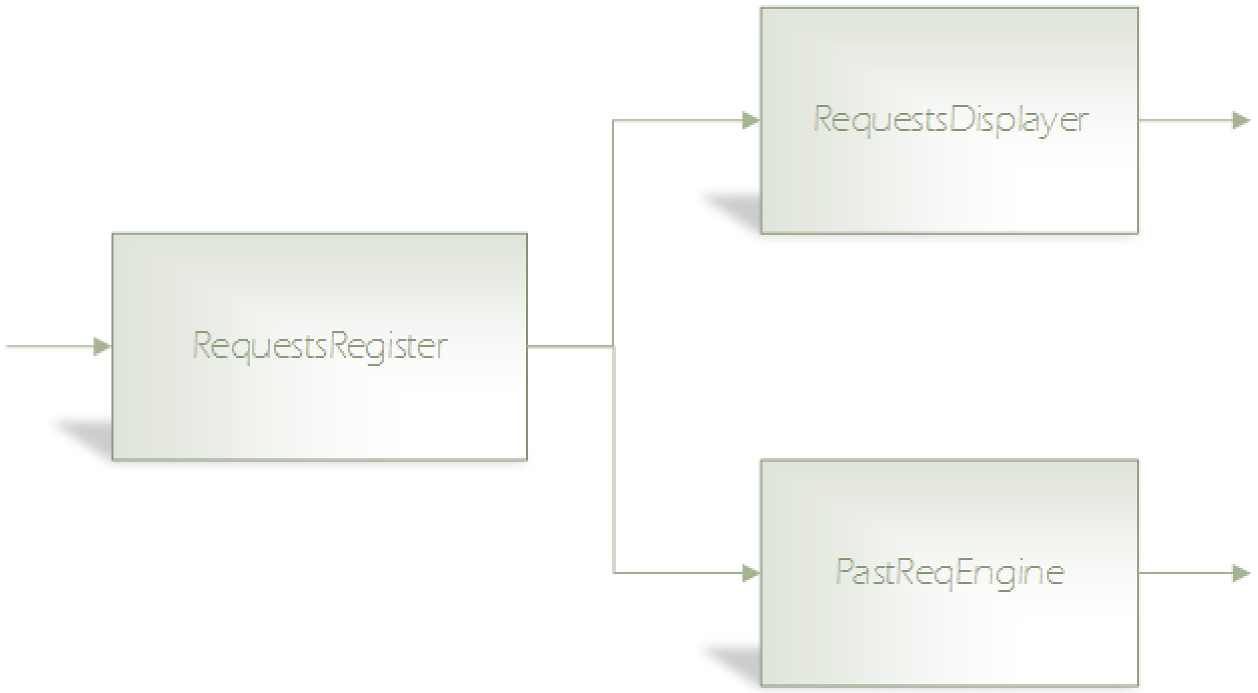
❖ ***Request processing***



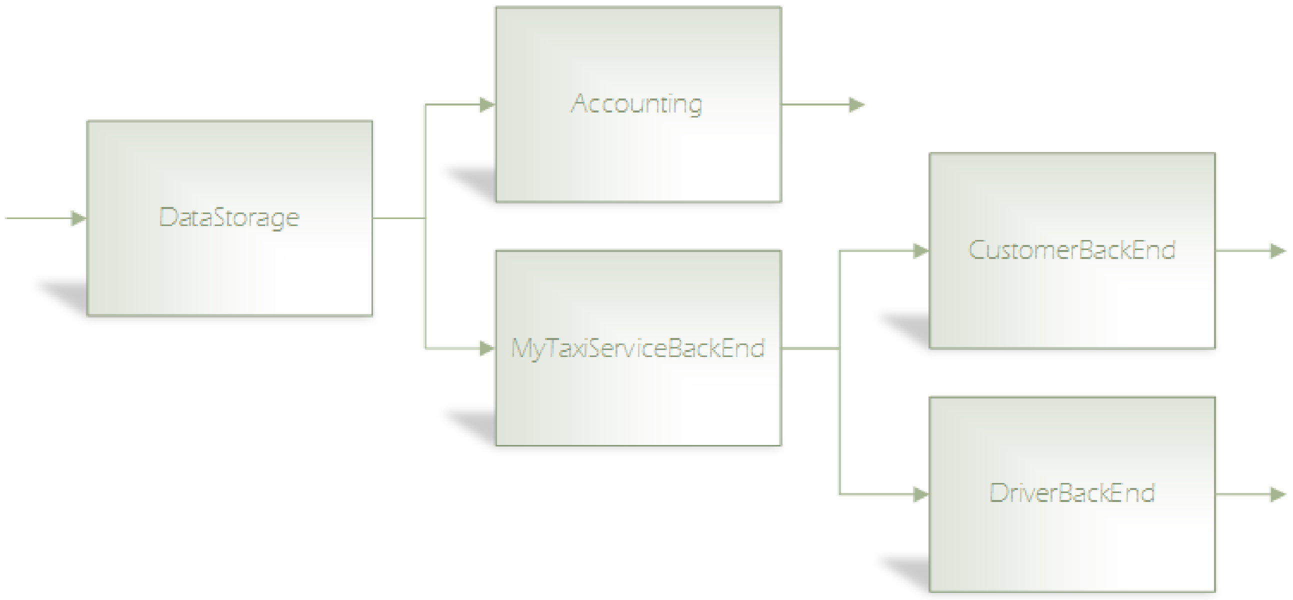
❖ ***Driver update state***



❖ ***Show past requests***



2.4.2 Subsystem Integration Sequence.



3. Individual Steps and Test Description

3.1 I1

Test Items(s)	AccountingWarehouse → AccountingManager
Input Specifications	Create the typical AccountingWarehouse input
Output Specifications	Check if the correct methods are called in AccountingManager
Environmental Needs	N/A
Purpose	This test verifies whether the AccountingManager can retreive the correct data on the database

3.2 I2

Test Items(s)	AccountingManager → CustomerBE
Input Specifications	Create the typical AccountingManager input
Output Specifications	Check if the correct methods are called in CustomerBE
Environmental Needs	I1 succeeded
Purpose	This test verifies whether the CustomerBE is correctly enabled

3.3 I3

Test Items(s)	AccountingManager → DriverBE
Input Specifications	Create the typical AccountingManager input
Output Specifications	Check if the correct methods are called in DriverBE
Environmental Needs	I1 succeeded
Purpose	This test verifies whether the DriverBE is correctly enabled

3.4 I4

Test Items(s)	AccountingWarehouse → QueueEngine
Input Specifications	Create the typical AccountingWarehouse input
Output Specifications	Check if the correct methods are called in QueueEngine
Environmental Needs	Request driver
Purpose	This test verifies whether the QueueEngine receives the correct data from the database

3.5 I5

Test Items(s)	QueueEngine → RequestHandler
Input Specifications	Create the typical QueueEngine input
Output Specifications	Check if the correct methods are called in RequestHandler
Environmental Needs	I4 succeeded
Purpose	This test verifies whether the QueueEngine can handle and output correctly the data

3.6 I6

Test Items(s)	NotificationEngine → NotificationDispatcher
Input Specifications	Create the typical NotificationEngine input
Output Specifications	Check if the correct methods are called in NotificationDispatcher
Environmental Needs	Request driver
Purpose	This test verifies whether the NotificationEngine can handle the request’s dispatched data

3.7 I7

Test Items(s)	NotificationDispatcher → RequestHandler
Input Specifications	Create the typical NotificationDIspatcher input
Output Specifications	Check if the correct methods are called in RequestHandler
Environmental Needs	I6 succeeded
Purpose	This test verifies whether the NotificationDispatcher can dispatch correctly information

3.8 I8

Test Items(s)	RequestEngine → RequestHandler
Input Specifications	Create the typical RequestEngine input
Output Specifications	Check if the correct methods are called in RequestHandler
Environmental Needs	Request driver
Purpose	This test verifies whether the RequestEngine can handle correctly input data

3.9 I9

Test Items(s)	RequestRegister → RequestHandler
Input Specifications	Create the typical RequestRegister input
Output Specifications	Check if the correct methods are called in RequestHandler
Environmental Needs	Request driver
Purpose	This test verifies whether data are correctly inserted in the database

3.10 I10

Test Items(s)	RequestHandler → RequestForwarder
Input Specifications	Create the typical RequestHandler input
Output Specifications	Check if the correct methods are called in RequestForwarder
Environmental Needs	I5, I7, I8, I9 succeeded
Purpose	This test verifies whether the RequestHandler can handle correctly request’s data

3.11 I11

Test Items(s)	RequestRegister → PastReqEngine
Input Specifications	Create the typical RequestRegister input
Output Specifications	Check if the correct methods are called in PastReqEngine
Environmental Needs	N/A
Purpose	This test verifies whether the PastReqEngine receives the correct data from the database

3.12 I12

Test Items(s)	RequestRegister → RequestDisplayer
Input Specifications	Create the typical RequestRegister input
Output Specifications	Check if the correct methods are called in RequestDisplayer
Environmental Needs	N/A
Purpose	This test verifies whether the RequestDisplayer receives the correct data from the database

3.13 I13

Test Items(s)	GpsLocalizer → AccountingWarehouse
Input Specifications	Create the typical data on GpsLocalizer
Output Specifications	Check if the correct methods are called in AccountingWarehouse
Environmental Needs	N/A
Purpose	This test verifies whether the data on the database are updated correctly

3.14 I14

Test Items(s)	AvailableSetter → AccountingWarehouse
Input Specifications	Create the typical data on AvailableSetter
Output Specifications	Check if the correct methods are called in AccountingWarehouse
Environmental Needs	N/A
Purpose	This test verifies whether the data on the database are updated correctly

4. Tools and Test Equipment Required

In order to perform the integrations test described in the previous sections we recommend to developers to use the following tools:

- *Mockito: this is a very useful tool, which allow creating mockups for unit testing. It permit a very easy creation of test which are more readable*
- *JUnit: this tool, used in cooperation with Mockito, allow writing unit tests; it has a lot of extensions and it makes easy to find solutions to problems*
- *Arquillian: this framework, used in cooperation with JUnit, allow ensuring the correct working of system functionalities, which are provided by the interaction between subsystems, also simulating the interaction with external clients*
- *Manual testing: it can be used to simulate customer experience*

5. Program Stubs and Test Data Required

Some program driver and special test data are required to correctly perform all the integration steps described above:

- *I4, I6, I8, and I9 need a request driver, in fact they represent the integration of the components which take care of all the functions carried out when a user make a request*
- *I1, I4, I13 and I14 need that in the AccountingWarehouse are inserted some fake Customer and Driver account to verify whether the interaction between the system and the database is properly done*
- *I11, I12 need that in the RequestRegister are inserted some fake request to verify whether the interaction between the system and the database is properly done*

6. Appendix

6.1 Tools

- *Microsoft Word: used for creating this document.*
- *Microsoft Visio: for creating the diagrams about the sequence of components and the function integration*
- *Microsoft OneDrive: use for sharing files among the group.*

6.2 Hours of work

- *Francesco: 10 hours.*
- *Marco: 10 hours.*