POLYTECHNIC UNIVERSITY OF MILAN

School of Industrial and Information Engineering

Computer Science and Engineering

Project of Software Engineering 2: MyTaxi Service
Integration Test Plan Document

Course Professor: Prof. Elisabetta DI NITTO

Authors:

Mattia CRIPPA 854126

Francesca GALLUZZI 788328

Marco LATTARULO 841399

Academic Year 2015–2016

Contents

Chapter 1

Introduction

Integration testing confirms that each piece of the application interacts as designed and that all functionality is working. Integration testing includes interactions between all layers of an application, including interfaces to other applications, as a complete end-to-end test of the functionality. The development team will be responsible for the creation of the integration test scripts in accordance to the integration test plan. A developer will be chosen by the team who will be responsible for execution of the test scripts and certifying that the integration testing is complete.

1.1 Revision History

Version 1.0, date 21/01/2016

1.2 Purpose

The purpose of the integration test plan is to describe the necessary tests to verify that all of the components of MyTaxiService are properly assembled. Integration testing ensures that the unit-tested modules interact correctly.

1.3 Scope

This document refers to the developing of an application called MyTaxiService, which is aimed to improve the quality and the efficiency of the taxi service of a large city by using localization, smartphones and IT technologies.

1.4 List of Definitions and Abbreviations

RASD: Requirement Analysis and Specification Document

ITPD: Integration Test Plan Document

IT: Integration Test

TP: Test Procedure

1.5 List of Reference Documents

List of all reference documents:

- Project Description: Assignments 1 and 2 (RASD and DD).pdf
- RASD: RASD v2.0-CrippaGalluzziLattarulo.pdf
- Design Document: DesignDocument v1.0-CrippaGalluzziLattarulo.pdf
- ITPD: Assignment 4 - integration test plan.pdf

1.6 Document Overview

This document is essentially structured in five parts:

- **Section 1** → Introduction: defines the purpose, the scope and an overview of this document.
- **Section 2** → Integration Strategy: defines all the items to be tested and the integration testing approach.
- **Section 6** → Individual Steps and Test Description: describes the type of test that will be used to verify that the elements perform as expected.
- **Section 4** → Tools and Test Equipment Required: defines all tools and test equipment needed to accomplish the integration.
- **Section 5** → Program Stubs and Test Data Required: defines any program stubs or special test data required.

Chapter 2

Integration Strategy

2.1 Entry Criteria

The Integration Testing can be carried out after the successful completion of the Unit Testing of the entire software. In addition the following points should be valid:

- The project should be code-complete and all its major features should be already present
- The project should satisfy the memory requirements specified in the RASD
- The correct version of the software is moved into the integration testing environment
- Sanity testing is done and build is stable for further testing
- The Database should be ready and its tables are populated with initial data

2.2 Elements to be Integrated

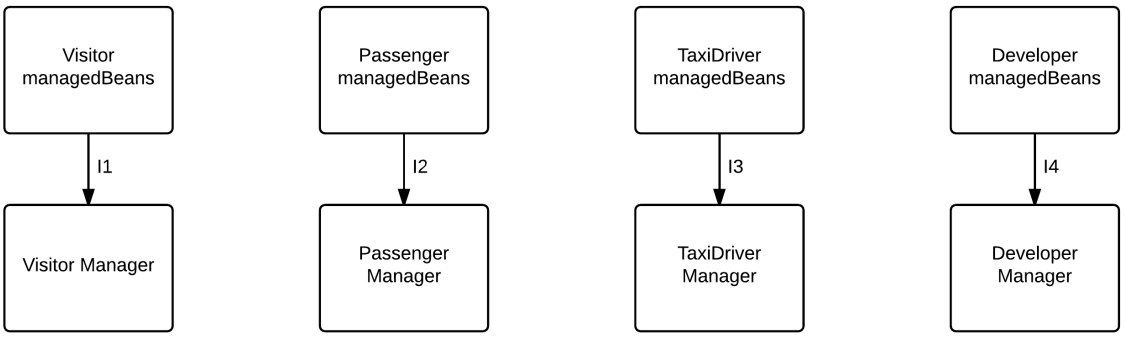
Due to the early stage of development of the software and the resulting low level of complexity of the entire system, we decided to focus our integration testing only on the main components of the Business Logic, keeping in mind that the future evolution of the project will lead to the creation of a number of subcomponent inside these component, needed to make the system fully working. Following this decision, we are going to integrate the Web Component and the Business Logic Component,

testing the direct connections between the managed Beans and their corresponding Managers and we are also going to integrate the 7 subcomponents of the Business Logic Component.

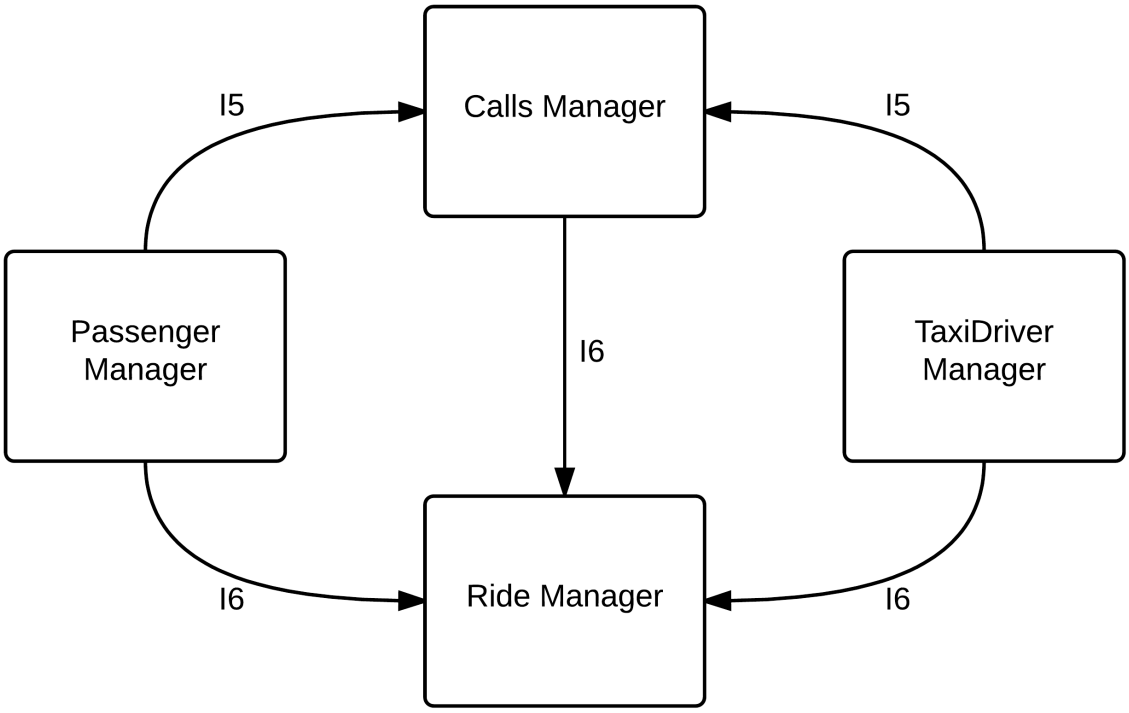
2.3 Integration Testing Strategy

Due to the particular stage of development explained in the previous point, at the moment there is not a complete hierarchy of (sub)components and (sub)systems, so it's not possible to fully define the integration test strategy followed in this document, because all the components involved are on the same level. Anyway the selected approach is the top-down approach, because (as stated in the previous point) in the future other lower level subcomponents will be implemented and then the testing will follow this downward development.

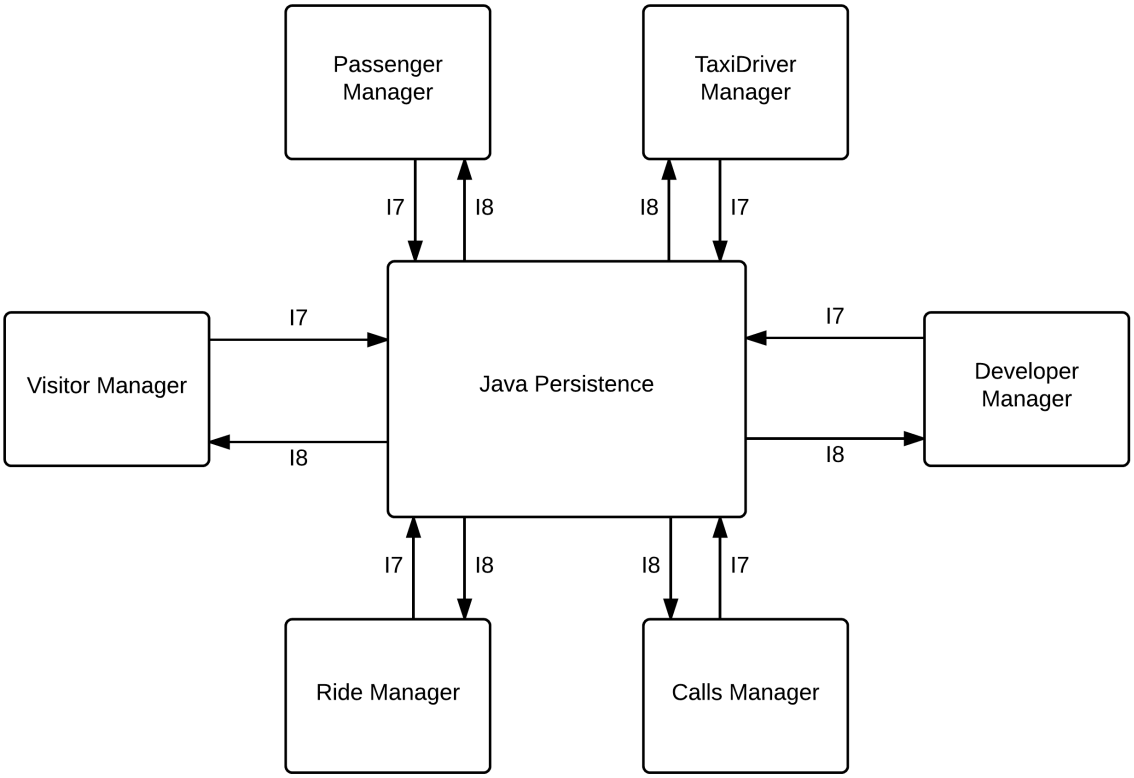
2.4 Sequence of Component / Function Integration



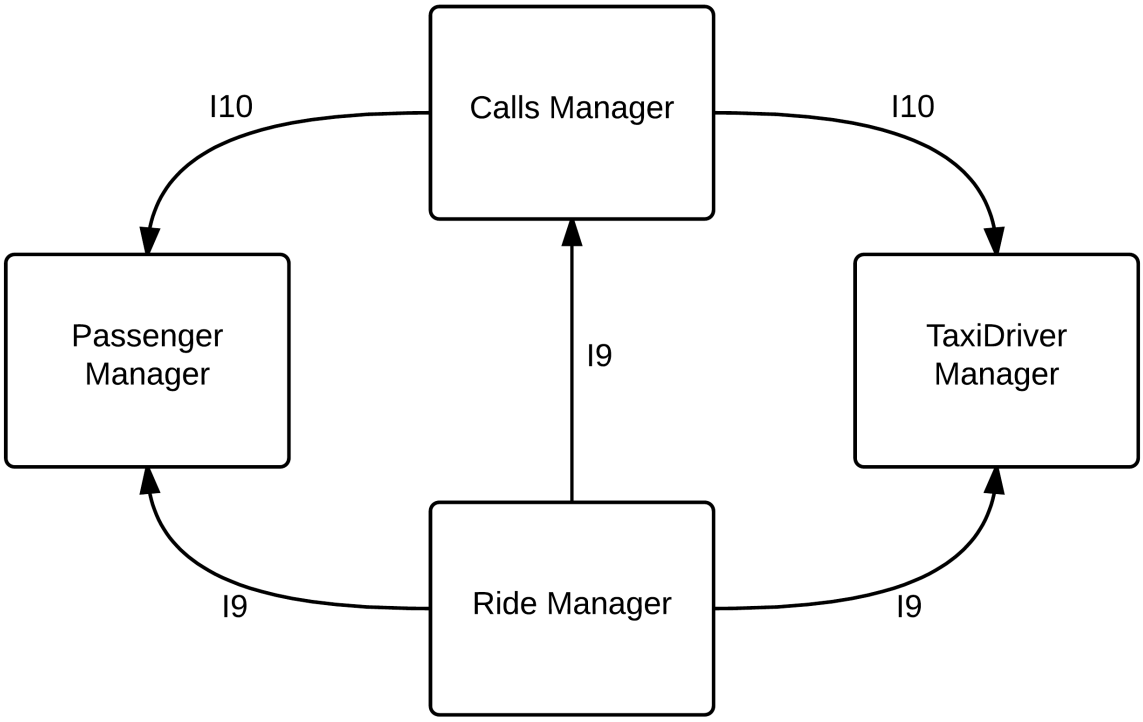
ID	Integration Test	Paragraphs
I1	Visitor managedBean → Visitor Manager	3.1.1 3.2.1
I2	Passenger managedBean → Passenger Manager	3.1.2 3.2.1
I3	TaxiDriver managedBean → TaxiDriver Manager	3.1.3 3.2.1
I4	Developer managedBean → Developer Manager	3.1.4 3.2.1



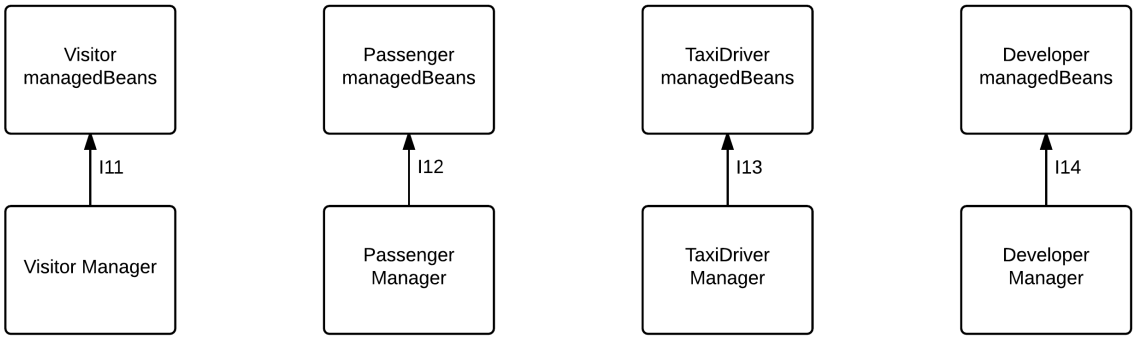
ID	Integration Test	Paragraphs
I5	Passenger Manager, TaxiDriver Manager → Calls Manager	3.1.5 3.2.2
I6	Passenger Manager, TaxiDriver Manager, Calls Manager → Ride Manager	3.1.6 3.2.2



ID	Integration Test	Paragraphs
I7	Visitor Manager, Passenger Manager, TaxiDriver Manager, Developer Manager, Calls Manager, Ride Manager → Java Persistence	3.1.7 3.2.3
I8	Java Persistence → Visitor Manager, Passenger Manager, TaxiDriver Manager, Developer Manager, Calls Manager, Ride Manager	3.1.8 3.2.3



ID	Integration Test	Paragraphs
I9	Ride Manager → Passenger Manager, TaxiDriver Manager, Calls Manager	3.1.9 3.2.4
I10	Calls Manager → Passenger Manager, TaxiDriver Manager	3.1.10 3.2.4



ID	Integration Test	Paragraphs
I11	Visitor Manager → Visitor managedBean	3.1.11 3.2.5
I12	Passenger Manager → Passenger managedBean	3.1.12 3.2.5
I13	TaxiDriver Manager → TaxiDriver managedBean	3.1.13 3.2.5
I14	Developer Manager → Developer managedBean	3.1.14 3.2.5

Chapter 3

Individual Steps & Test Description

3.1 Test case specifications

3.1.1 Integration test case I1

Test Case Identifier	I1T1
Test Item	Visitor managedBeans → Visitor Manager
Input Specification	Create typical Visitor managedBeans input
Output Specification	Check if the correct methods are called in the Visitor Manager
Environmental Needs	Client driver

3.1.2 Integration test case I2

Test Case Identifier	I2T1
Test Item	Passenger managedBeans → Passenger Manager
Input Specification	Create typical Passenger managedBeans input
Output Specification	Check if the correct methods are called in the Passenger Manager
Environmental Needs	Client driver

3.1.3 Integration test case I3

Test Case Identifier	I3T1
Test Item	TaxiDriver managedBeans → TaxiDriver Manager
Input Specification	Create typical TaxiDriver managedBeans input
Output Specification	Check if the correct methods are called in the TaxiDriver Manager
Environmental Needs	Client driver

3.1.4 Integration test case I4

Test Case Identifier	I4T1
Test Item	Developer managedBeans → Developer Manager
Input Specification	Create typical Developer managedBeans input
Output Specification	Check if the correct methods are called in the Developer Manager
Environmental Needs	Client driver

3.1.5 Integration test case I5

Test Case Identifier	I5T1
Test Item	Passenger Manager → Calls Manager
Input Specification	Create typical Passenger Manager input
Output Specification	Check if the correct methods are called in the Calls Manager
Environmental Needs	I2 succeded

Test Case Identifier	I5T2
Test Item	TaxiDriver Manager → Calls Manager
Input Specification	Create typical TaxiDriver Manager input
Output Specification	Check if the correct methods are called in the Calls Manager
Environmental Needs	I3 succeded

3.1.6 Integration test case I6

Test Case Identifier	I6T1
Test Item	Passenger Manager → Ride Manager
Input Specification	Create typical Passenger Manager input
Output Specification	Check if the correct methods are called in the Ride Manager
Environmental Needs	I2 succeded

Test Case Identifier	I6T2
Test Item	TaxiDriver Manager → Ride Manager
Input Specification	Create typical TaxiDriver Manager input
Output Specification	Check if the correct methods are called in the Ride Manager
Environmental Needs	I3 succeded

Test Case Identifier	I6T3
Test Item	Calls Manager → Ride Manager
Input Specification	Create typical Calls Manager input
Output Specification	Check if the correct methods are called in the Ride Manager
Environmental Needs	I2, I3, I5 succeded

3.1.7 Integration test case I7

Test Case Identifier	I7T1
Test Item	Visitor Manager → Java Persistence
Input Specification	Create typical Visitor Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I1 succeded

Test Case Identifier	I7T2
Test Item	Passenger Manager → Java Persistence
Input Specification	Create typical Passenger Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I2 succeded

Test Case Identifier	I7T3
Test Item	TaxiDriver Manager → Java Persistence
Input Specification	Create typical TaxiDriver Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I3 succeded

Test Case Identifier	I7T4
Test Item	Developer Manager → Java Persistence
Input Specification	Create typical Developer Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I4 succeded

Test Case Identifier	I7T5
Test Item	Calls Manager → Java Persistence
Input Specification	Create typical Calls Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I2, I3 and I5 succeded

Test Case Identifier	I7T6
Test Item	Ride Manager → Java Persistence
Input Specification	Create typical Ride Manager input
Output Specification	Check if the correct methods are called in the Persistence Module
Environmental Needs	I2, I3, I5 and I6 succeded

3.1.8 Integration test case I8

Test Case Identifier	I8T1
Test Item	Java Persistence → Visitor Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the Visitor Manager
Environmental Needs	Database Driver

Test Case Identifier	I8T2
Test Item	Java Persistence → Passenger Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the Passenger Manager
Environmental Needs	Database Driver

Test Case Identifier	I8T3
Test Item	Java Persistence → TaxiDriver Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the TaxiDriver Manager
Environmental Needs	Database Driver

Test Case Identifier	I8T4
Test Item	Java Persistence → Developer Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the Developer Manager
Environmental Needs	Database Driver

Test Case Identifier	I8T5
Test Item	Java Persistence → Calls Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the Calls Manager
Environmental Needs	Database Driver

Test Case Identifier	I8T6
Test Item	Java Persistence → Ride Manager
Input Specification	Create typical Java Persistence input
Output Specification	Check if the correct methods are called in the Ride Manager
Environmental Needs	Database Driver

3.1.9 Integration test case I9

Test Case Identifier	I9T1
Test Item	Ride Manager → Passenger Manager
Input Specification	Create typical Ride Manager input
Output Specification	Check if the correct methods are called in the Passenger Manager
Environmental Needs	I8 succeded

Test Case Identifier	I9T2
Test Item	Ride Manager → TaxiDriver Manager
Input Specification	Create typical Ride Manager input
Output Specification	Check if the correct methods are called in the TaxiDriver Manager
Environmental Needs	I8 succeded

Test Case Identifier	I9T3
Test Item	Ride Manager → Calls Manager
Input Specification	Create typical Ride Manager input
Output Specification	Check if the correct methods are called in the Calls Manager
Environmental Needs	I8 succeded

3.1.10 Integration test case I10

Test Case Identifier	I10T1
Test Item	Calls Manager → Passenger Manager
Input Specification	Create typical Calls Manager input
Output Specification	Check if the correct methods are called in the Passenger Manager
Environmental Needs	I8 and I9 succeded

Test Case Identifier	I10T2
Test Item	Calls Manager → TaxiDriver Manager
Input Specification	Create typical Calls Manager input
Output Specification	Check if the correct methods are called in the TaxiDriver Manager
Environmental Needs	I8 and I9 succeded

3.1.11 Integration test case I11

Test Case Identifier	I11T1
Test Item	Visitor Manager → Visitor managedBeans
Input Specification	Create typical Visitor Manager input
Output Specification	Check if the correct methods are called in the Visitor managedBeans
Environmental Needs	I8 succeded

3.1.12 Integration test case I12

Test Case Identifier	I12T1
Test Item	Passenger Manager → Passenger managedBeans
Input Specification	Create typical Passenger Manager input
Output Specification	Check if the correct methods are called in the Passenger managedBeans
Environmental Needs	I8, I9 and I10 succeded

3.1.13 Integration test case I13

Test Case Identifier	I13T1
Test Item	TaxiDriver Manager → TaxiDriver managedBeans
Input Specification	Create typical TaxiDriver Manager input
Output Specification	Check if the correct methods are called in the TaxiDriver managedBeans
Environmental Needs	I8, I9 and I10 succeded

3.1.14 Integration test case I14

Test Case Identifier	I14T1
Test Item	Developer Manager → Developer managedBeans
Input Specification	Create typical Developer Manager input
Output Specification	Check if the correct methods are called in the Developer managedBeans
Environmental Needs	I8 succeded

3.2 Test procedures

3.2.1 Test procedure TP1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether all the Managers of the Business Layer can successfully: • receive requests from the corresponding managed-Beans • correctly elaborate those requests
Procedure Steps	Execute I1, I2, I3, I4

3.2.2 Test procedure TP2

Test Procedure Identifier	TP2
Purpose	This test procedure verifies whether the Calls Manager and the Ride Manager can successfully receive and handle requests from: • Visitor Manager • Passenger Manager • TaxiDriver Manager • Developer Manager Also, this procedure verifies whether the Ride Manager can successfully receive and handle requests from the Calls Manager
Procedure Steps	Execute I5 and I6

3.2.3 Test procedure TP3

Test Procedure Identifier	TP3
Purpose	This test procedure verifies whether the Java Persistence Module can successfully receive, handle and reply to requests from: • Visitor Manager • Passenger Manager • TaxiDriver Manager • Developer Manager • Calls Manager • Ride Manager
Procedure Steps	Execute I8 after I7

3.2.4 Test procedure TP4

Test Procedure Identifier	TP4
Purpose	This test procedure verifies whether the Passenger Manager and the TaxiDriver Manager can successfully receive and handle inputs from: • Calls Manager • Ride Manager
Procedure Steps	Also, this procedure verifies whether the Calls Manager can successfully receive and handle inputs from Ride Manager Execute I9 and I10

3.2.5 Test procedure TP5

Test Procedure Identifier	TP5
Purpose	This test procedure verifies whether all the managed-Beans can successfully: • receive inputs from the corresponding Managers • correctly elaborate those inputs
Procedure Steps	Execute I11, I12, I13, I14

Chapter 4

Tools & Test Equipment Required

For carrying out the Integration Test we decided to use *Arquillian*, an integration testing framework for Java EE, because it allows to make integration test as simple to write as unit test. Additional test equipment required in order to perform integration test are the GPS receivers specified in the RASD and a smartphone with characteristics that respect all the requirements already defined in RASD itself.

Chapter 5

Program Stubs & Test Data Required

5.1 Program Stubs

All the client's inputs will be simulated by specific drivers. There sholud be at least 4 such drivers, one for each type of user of the system.

The Visitor Driver should simulate these inputs:

- creation of a new user (by filling the registration form)
- login with proper username and password

The Passenger Driver should simulate these inputs:

- request of a Taxi
- reservation of a Taxi
- visualization of the Account Page and modification of some personal information
- visualization of a receipt
- visualization of the Home Page

The TaxiDriver Driver should simulate these inputs:

- visualization of the Dashboard and interaction with the route and with the incoming calls

- visualization of the Summary Page
- visualization of the Account Page and modification of some personal information
- visualization of the Home Page

The Developer Driver should simulate these inputs:

- visualization of the Testing Page
- visualization of technical information
- insertion of new features or modification of some features

5.2 Test Data

For successfully carrying out the test procedures, all the following tables of the Database must be populated:

- User
- Developer
- Passenger
- TaxiDriver
- Passenger Ride
- Taxi Ride
- City Zone
- Taxicab
- GPS Receiver

Furthermore, a dataset of location data to simulate inputs from the GPS Receiver and from the smartphones is needed.

Chapter 6

Other Information

6.1 Working Hours

First Name	Last Name	Total Hours
Mattia	Crippa	16h
Francesca	Galluzzi	16h
Marco	Lattarulo	17h