Politecnico di Milano

Integration Test Plan Document

"myTaxiService"

Nicolas Tagliabue(matr. 853097), Matteo Pagliari(matr. 854353)

January 21, 2016

Versione 1.0

Contents

1 Introduction

1.1 Purpose

The purpose of the Integration Testing document is to describe how the components in our system will be integrated for the testing phase according the design choices made. The testing will verify that all the functional requirements described in the previous documents are met, but also that every single subsystem is working as it should. It will also be verified that the right sequence of software components is called in order to accomplish the desired task.

1.2 Scope

This document is intended for the development of myTaxiService which is a web and mobile application intended to implement an easier way to coordinate the taxi system of a particular city. Its focus is on the testing part of the devolpment flow.

1.3 Definitions, Acronyms, Abbreviations

For a metter of shortness, when referring to the components the word "Manager" has been omitted

1.4 Reference Documents

See:

- Project Description
- RASD
- DD
- Junit (http://junit.org/)
- Mockito (http://site.mockito.org/)
- Arquillan (http://arquillian.org/)

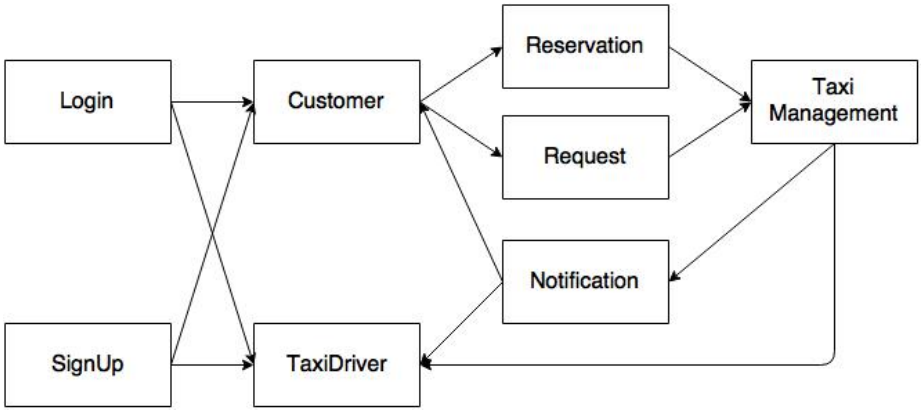
2 Integration Strategy

2.1 Entry Criteria

The main point is that every component that we are going to test now should have been tested before, as a single component and proven that is working correctly (Unit Testing). As a reference, in the SDD the is a description of all the component that will be integrated here.

2.2 Elements to be Integrated

Here is a picture indicating all the elements that will be tested



2.3 Integration Testing Strategy

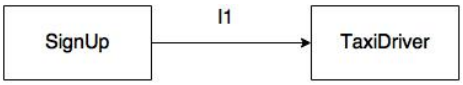
Considering the structure of the system we decided to adopt a functional grouping way of testing because it seemed more logical for us to test the different subsystems and components according to their similarities of functional behaviour. In this way we will test each functionality individually, in order to consider them together later on. According to the previous documents, we have identified 5 different groups, each one of them corresponding to a different functionality: SignUp, Login, Request, Reservation, Notification.

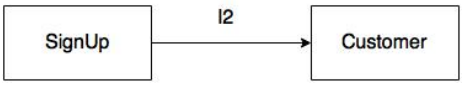
2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

Sign Up

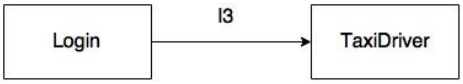
SignUp	
I1	SignUp -> TaxiDriver
I2	SignUp -> Customer

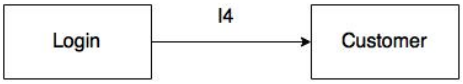




Log In

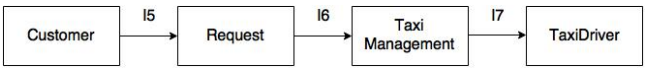
Login	
I3	Login -> TaxiDriver
I4	Login -> Customer





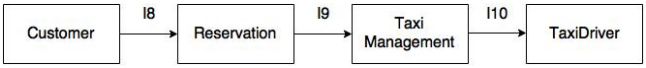
Request

Requests	
I5	Customer -> Request
I6	Request -> TaxiManagement
I7	TaxiManagement -> TaxiDriver



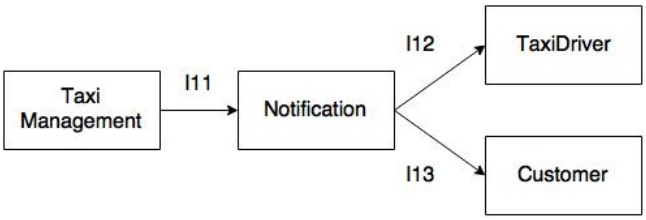
Reservations

Reservations	
I8	Customer -> Reservation
I9	Reservation-> TaxiManagement
I10	TaxiManagement -> TaxiDriver



Notification

Notification	
I11	TaxiManagement -> Notification
I12	Notification -> TaxiDriver
I13	Notification -> Customer



2.4.2 Subsystem Integration Sequence

Login - Request/Reservations	
T1	Login -> Request
T2	Login -> Reservation
Request/Reservation - Notification	
T3	Request/Reservation -> Notification

3 Individual Steps and Test Data Required

3.1 Integration Test case I1

Test Case Identifier	I1T1
Test Items	SignUp -> TaxiDriver
Input Specification	Registration data
Output Specification	Check if a TaxiDriver is created in case of correct input, otherwise no user is created.
Environmental Needs	Taxi Driver driver

3.2 Integration Test case I2

Test Case Identifier	I2T1
Test Items	SignUp -> Customer
Input Specification	Registration data
Output Specification	Check if a Customer is created in case of correct input, otherwise no user is created.
Environmental Needs	Customer driver

3.3 Integration Test case I3

Test Case Identifier	I3T1
Test Items	LogIn -> TaxiDriver
Input Specification	Authentication data
Output Specification	Check if the user is logged as TaxiDriver in case of valid input, otherwise no.
Environmental Needs	Taxi Driver driver

3.4 Integration Test case I4

Test Case Identifier	I4T1
Test Items	LogIn -> Customer
Input Specification	Authentication data
Output Specification	Check if the user is logged as Customer in case of valid input, otherwise no.
Environmental Needs	Customer driver

3.5 Integration Test case I5

Test Case Identifier	I5T1
Test Items	Customer -> Request
Input Specification	Request data
Output Specification	Check if the Customer provided info are valid, otherwise no.

3.6 Integration Test case I6

Test Case Identifier	I6T1
Test Items	Request -> TaxiManagement
Input Specification	Customer position
Output Specification	Check if a TaxiDriver in the correct zone is selected in case of valid input, otherwise no.

3.7 Integration Test case I7

Test Case Identifier	I7T1
Test Items	TaxiManagement -> TaxiDriver
Input Specification	Typical TaxiDriver input
Output Specification	Check if the correct functions are called in the TaxiDriver

3.8 Integration Test case I8

Test Case Identifier	I8T2
Test Items	Customer -> Reservation
Input Specification	Request data
Output Specification	Check if the Customer provided info are valid, otherwise no.

3.9 Integration Test case I9

Test Case Identifier	I9T2
Test Items	Reservation -> TaxiManagement
Input Specification	Customer position and destination of the route
Output Specification	Check if a TaxiDriver in the correct zone is selected in case of valid input, otherwise no.

3.10 Integration Test case I10

Test Case Identifier	I10T2
Test Items	TaxiManagement -> TaxiDriver
Input Specification	Typical TaxiDriver input
Output Specification	Check if the correct functions are called in the TaxiDriver

3.11 Integration Test case I11

Test Case Identifier	I11T3
Test Items	TaxiManagement -> Notification
Input Specification	Request/Reservation data
Output Specification	Check if a Notification is created in case of correct input, otherwise no.

3.12 Integration Test case I12

Test Case Identifier	I12T3
Test Items	Notification -> TaxiDriver
Input Specification	Notification data
Output Specification	Check if the Notification is send to the correct TaxiDriver.

3.13 Integration Test case I13

Test Case Identifier	I13T3
Test Items	Notification -> Customer
Input Specification	Notification data
Output Specification	Check if the Notification is send to the Customer who created the request.

3.14 Integration Test procedure T1

Test Procedure Identifier	T1
Purpose	This test procedure verifies that Request subsystems: - can manage Customer input - can manage TaxiDriver input - allows request by logged customer
Procedure steps	Execute I5-I7 after I1-I4.

3.15 Integration Test procedure T2

Test Procedure Identifier	T2
Purpose	This test procedure verifies that Reservation subsystems: - can manage Customer input - can manage TaxiDriver input - allows request by logged customer
Procedure steps	Execute I8-I10 after I1-I4.

3.16 Integration Test procedure T3

Test Procedure Identifier	T3
Purpose	This test procedure verifies that Notification subsystems: - can manage TaxiManagement input - can send notification to customers and taxidrivers involved in a Request/Reservation.
Procedure steps	Execute I11-I13 after I5-I10.

4 Tools and Test

For the testing phase we decided to use three different tools:

- Mockito (an open source testing framework for Java that allows the creation of test double objects (mock objects) in automated unit tests)
- Arquillan (an innovative and highly extensible testing platform for the JVM that enables developers to easily create automated integration, functional and acceptance tests for Java)
- Junit (JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks)

5 Program Stubs and Test Data Required

We need two different drivers for testing the login and signup functionalities: the TaxiDriver driver and the Customer driver.

We also need one stub, which is when a customer does a request, the Request manager calls the method checkValidPosition() that returns a boolean variable that represents if the origin position is correct.