Politecnico di Milano
A.Y. 2015/2016

myTaxiService

Integration Test Plan Document

Bernardis Cesare matr. 852509 Dagrada Mattia matr.852975

January 21, 2016

Contents

1 Introduction

1.1 Revision History

- Version 1.0 (Actual)

1.2 Purpose and Scope

The Test Plan Document (ITPD) aims at describing how you plan to accomplish the integration test. This document is supposed to be written before the integration test really happens. Often it is written in parallel to the Design Document and takes the architectural description of the software system as a starting point. This document needs to explain to the development team what to test, in which sequence, which tools are needed for testing (if any), which stubs/ drivers/oracles need to be developed.

1.3 List of Definitions and Abbreviations

1.4 List of Reference Documents

- myTaxiService Project description
- myTaxiService Requirement Analysis and Specification Document
- myTaxiService Design Document
- myTaxiService Requirement Analysis and Specification Document
- Assignment 4 - integration test plan

2 Integration Strategy

2.1 Entry Criteria

To proceed at the Integration Testing level treated in this document is necessary that every component of the system has been previously unit tested with a positive result. We must be sure that all the methods inside the components, here considered like black boxes, work as expected. This is a necessary constraint that must be satisfied, because the purpose of the tests discussed in this document is to check the interaction between components (that's why it is called "integration testing").

2.2 Elements to be Integrated

Premise: for a more detailed description of the system structure, refer to myTaxiService Design Document, sections 2.3 and 2.7.

Our system has been originally divided in four subsystems:

- Client
- Frontend System
- Backend System
- Database

Every subsystem is composed by a variable number of components and every single component has been unit tested individually. In this document we want to test the interaction between different components, so we are going to test all the interfaces provided by the components, dedicating a test case to every different interaction that involves an interface. Practically we want to test all the couples of components that interact between them.

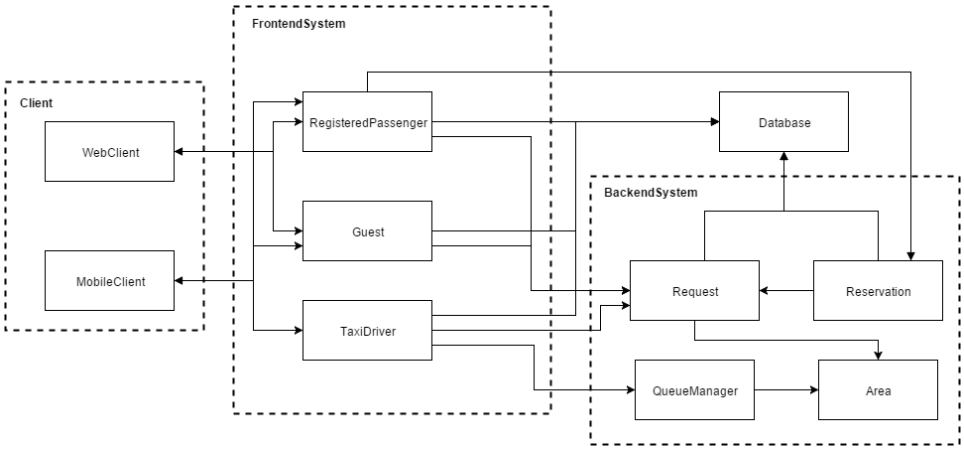

Figure 1: A particular component view

2.3 Integration Testing Strategy

We have chosen a bottom-up strategy. We think it's the best choice because represents a more gradual approach to the system testing. It also allows to find an eventual issue at the deepest level of integration, giving the possibility to intervene immediately at the correct level, without having to resort to a tougher analysis.

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

The order of the tests into the tables (and the growing ID number too) represents also the sequence in which the components will be integrated within each subsystem.

2.4.1.1 Integration Test of the Database - software

ID	Integration Test
I1	Request → Database
I2	Reservation → Database
I3	Guest → Database
I4	RegisteredPassenger → Database
I5	TaxiDriver → Database

2.4.1.2 Integration Test of the Backend system - software

ID	Integration Test
I6	Request → Area
I7	Guest → Request
I8	RegisteredPassenger → Request
I9	RegisteredPassenger → Reservation
I10	TaxiDriver → Request
I11	TaxiDriver → QueueManager

2.4.1.3 Integration Test of the Frontend system - software

ID	Integration Test
I12	Client → Guest
I13	Client → RegisteredPassenger
I14	MobileClient → TaxiDriver

2.4.1.4 Integration Test of the UserClient - software

ID	Integration Test
I15	MobileClient ↔ System
I16	WebClient ↔ System

2.4.2 Subsystem Integration Sequence

When all the subsystems have been individually tested, the can be integrated one by one attaching, the subsystems sequentially in the following order:

Order	Subsystem
1	Database
2	BackendSystem
3	FrontendSystem
4	Client

When the integration testing reaches the top level and the whole system has been integrated, some integration test procedures can be individuated analysing the main functionalities that our system has to provide.

Test Procedure ID	TP1
Purpose	This test procedure verifies whether the registration process works: • can access both the Clients • can fill the registration form • can store a RegisteredPassenger
Procedure Steps	I4T1, I12T1, I15T1 and I16T1(MobileClient), I17T1 and I18T1(WebClient)
Test Procedure ID	TP2
Purpose	This test procedure verifies whether the login procedure works for the RegisteredPassenger: • can access both the Clients • can fill the login form • can retrieve the RegisteredPassenger
Procedure Steps	I4T2, I12T2, I15T1 and I16T1(MobileClient), I17T1 and I18T1(WebClient)

Test Procedure ID	TP3
Purpose	This test procedure verifies whether the login procedure works for the TaxiDriver and he is able to manage his availability: • can access the MobileClient • can fill the login form • can retrieve the TaxiDriver • can give his availability • can revoke his availability
Procedure Steps	I5T2, I14T1, I11T2, I14T4, I11T1, I14T5, I15T1 and I16T1(MobileClient)

Test Procedure ID	TP4
Purpose	This test procedure verifies whether a Guest can make a request: • can fill the request form • can create and store the request • can submit the request to a TaxiDriver
Procedure Steps	I1T1, I3T2, I5T2, I6T1, I7T1, I12T3

Test Procedure ID	TP5
Purpose	This test procedure verifies whether a RegisteredPassenger can make a request: • can create and store the request • can submit the request to a TaxiDriver
Procedure Steps I1T1, I4T2, I5T2, I6T1, I8T1, I13T1	

Test Procedure ID	TP6
Purpose	This test procedure verifies whether a TaxiDriver can answer to a request: • can view the notification of a request • can answer positively • can answer negatively
Procedure Steps I5T2, I7T1, I8T1, I10T1(positive). I10T2(negative)	

Test Procedure ID	TP7
Purpose	This test procedure verifies whether a RegisteredPassenger can make a reservation: • can fill the reservation form • can create and store the reservation
Procedure Steps	I2T1, I8T1, I9T1, I13T2

Test Procedure ID	TP8
Purpose	This test procedure verifies whether a RegisteredPassenger can delete a reservation: • can view his reservations • can delete the selected reservation
Procedure Steps	I2T3, I2T2, I13T3

3 Individual Steps and Test Description

3.1 Integration Tests

3.1.1 Integration test case I1

Test Case Identifier	I1T1
Test Case Description	Store a Request into the database
Test Item(s)	Request → Database
Input Specification	A Request component data
Output Specification	Check that the correct method is called and that the data stored in the Database matches with the input data
Environmental Needs	None

3.1.2 Integration test case I2

Test Case Identifier	I2T1
Test Case Description	Store a Reservation into the database
Test Item(s)	Reservation → Database
Input Specification	A Reservation component data
Output Specification	Check that the correct method is called and that the data stored in the Database matches with the input data
Environmental Needs	None

Test Case Identifier	I2T2
Test Case Description	Delete a Reservation from the database
Test Item(s)	Reservation → Database
Input Specification	A Reservation identifier
Output Specification	Check that the correct method is called and that the data stored in the Database that matches the input identifier is deleted correctly. If there are no matches nothing is changed
Environmental Needs	None

Test Case Identifier	I2T3
Test Case Description	Retreive a Reservation from the database
Test Item(s)	Reservation → Database
Input Specification	A Reservation identifier
Output Specification	Check that the correct method is called and that the object returned matches with the data stored into the database
Environmental Needs	None

3.1.3 Integration test case I3

Test Case Identifier	I3T1
Test Case Description	Store a Guest into the database
Test Item(s)	Guest → Database
Input Specification	A Guest component data
Output Specification	Check that the correct method is called and that the data stored in the Database matches with the input data
Environmental Needs	None

Test Case Identifier	I3T2
Test Case Description	Retreive a Guest from the database
Test Item(s)	Guest → Database
Input Specification	A Guest identifier
Output Specification	Check that the correct method is called and that the object returned matches with the data stored into the database
Environmental Needs	None

3.1.4 Integration test case I4

Test Case Identifier	I4T1
Test Case Description	Store a RegisteredPassenger into the database
Test Item(s)	RegisteredPassenger → Database
Input Specification	A RegisteredPassenger component data
Output Specification	Check that the correct method is called and that the data stored in the Database matches with the input data
Environmental Needs	None

Test Case Identifier	I4T2
Test Case Description	Retreive a RegisteredPassenger from the database
Test Item(s)	RegisteredPassenger → Database
Input Specification	A RegisteredPassenger identifier
Output Specification	Check that the correct method is called and that the object returned matches with the data stored into the database
Environmental Needs	None

3.1.5 Integration test case I5

Test Case Identifier	I5T1
Test Case Description	Store a TaxiDriver into the database
Test Item(s)	TaxiDriver → Database
Input Specification	A TaxiDriver component data
Output Specification	Check that the correct method is called and that the data stored in the Database matches with the input data
Environmental Needs	None

Test Case Identifier	I5T2
Test Case Description	Retreive a TaxiDriver from the database
Test Item(s)	TaxiDriver → Database
Input Specification	A TaxiDriver identifier
Output Specification	Check that the correct method is called and that the object returned matches with the data stored into the database
Environmental Needs	None

3.1.6 Integration test case I6

Test Case Identifier	I6T1
Test Case Description	Retrieve the head of the queue af an area
Test Item(s)	Request → Area
Input Specification	None
Output Specification	Check that the correct method is called, that the TaxiDriver returned was the one on the top of the queue and that the queue has not been touched (head excluded)
Environmental Needs	Request Driver

3.1.7 Integration test case I7

Test Case Identifier	I7T1
Test Case Description	A guest makes a request
Test Item(s)	Guest → Request
Input Specification	A Guest component data
Output Specification	Check that the correct method is called and that the component returned matches with the input data
Environmental Needs	Guest Driver, I1T1, I3T2, I5T2, I6T1

3.1.8 Integration test case I8

Test Case Identifier	I8T1
Test Case Description	A logged user makes a request
Test Item(s)	RegisteredPassenger → Request
Input Specification	A RegisteredPassenger component data
Output Specification	Check that the correct method is called and that the component returned matches with the input data
Environmental Needs	RegisteredPassenger Driver, I1T1, I4T2, I5T2, I6T1

3.1.9 Integration test case I9

Test Case Identifier	I9T1
Test Case Description	A logged user makes a reservation
Test Item(s)	RegisteredPassenger → Reservation
Input Specification	A RegisteredPassenger component data, a time and a place
Output Specification	Check that the correct method is called and that the component returned matches with the input data
Environmental Needs	I2T1, I8T1

3.1.10 Integration test case I10

Test Case Identifier	I10T1
Test Case Description	Taxi driver answers positively to a request
Test Item(s)	TaxiDriver → Request
Input Specification	The Request and a positive answer
Output Specification	Check that the correct method is called and that the answer is recorded into the system, satisfying the request.
Environmental Needs	TaxiDriver Driver, I5T2, I7T1, I8T1

Test Case Identifier	I10T2
Test Case Description	Taxi driver answers negatively to a request
Test Item(s)	TaxiDriver → Request
Input Specification	The Request and negative answer
Output Specification	Check that the correct method is called and that a new TaxiDriver is removed from his queue to be interpellated
Environmental Needs	TaxiDriver Driver, I5T2, I7T1, I8T1

3.1.11 Integration test case I11

Test Case Identifier	I11T1
Test Case Description	Remove a taxi driver from a queue
Test Item(s)	TaxiDriver → QueueManager
Input Specification	A TaxiDriver component data
Output Specification	Check that the correct method is called and that the TaxiDriver is correctly removed without changing the order of the other TaxiDrivers in the queue
Environmental Needs	TaxiDriver driver

Test Case Identifier	I11T2
Test Case Description	Add a taxi driver to a queue
Test Item(s)	TaxiDriver → QueueManager
Input Specification	A TaxiDriver component data
Output Specification	Check that the correct method is called and that the TaxiDriver is correctly inserted at the end of his area queue
Environmental Needs	TaxiDriver driver

3.1.12 Integration test case I12

Test Case Identifier	I12T1
Test Case Description	A Guest performs a registration
Test Item(s)	Client → Guest
Input Specification	All the personal data required to fill the registration form
Output Specification	If the registration has been successful a success notification is shown, if not an error notification is shown
Environmental Needs	I4T1

Test Case Identifier	I12T2
Test Case Description	A Guest logs into the service
Test Item(s)	Client → Guest
Input Specification	The required log in data
Output Specification	If the log in has been successful, the Client is now logged in as a RegisteredPassenger, if not an error notification is shown
Environmental Needs	I4T2

Test Case Identifier	I12T3
Test Case Description	A Guest makes a request
Test Item(s)	Client → Guest
Input Specification	Personal data required in order to make a request and the position consensus
Output Specification	There is no direct output but the Client will receive a notification if the request has been successful
Environmental Needs	I7T1

3.1.13 Integration test case I13

Test Case Identifier	I13T1
Test Case Description	A RegisteredPassenger makes a request
Test Item(s)	Client → RegisteredPassenger
Input Specification	Nothing, all the data about the RegisteredPassenger are already store into the database. He will have to give the position consensus
Output Specification	There is no direct output but the Client will receive a notification if the request has been successful
Environmental Needs	I8T1

Test Case Identifier	I13T2
Test Case Description	A RegisteredPassenger makes a reservation
Test Item(s)	Client → RegisteredPassenger
Input Specification	The origin and the data with the time
Output Specification	There is no direct output but the Client will receive a notification if the reservation has been successful. He will also receive a notification once the request related to the reservation starts
Environmental Needs	I9T1

Test Case Identifier	I13T3
Test Case Description	A RegisteredPassenger deletes a reservation
Test Item(s)	Client → RegisteredPassenger
Input Specification	The selected reservation
Output Specification	There is no direct output but the Client will see that the reservation is no longer there
Environmental Needs	I2T3, I2T2

3.1.14 Integration test case I14

Test Case Identifier	I14T1
Test Case Description	A TaxiDriver logs into the service
Test Item(s)	MobileClient → TaxiDriver
Input Specification	The required log in data
Output Specification	If the log in has been successful, so the Client is now logged in as a TaxiDriver, if not an error notification is shown
Environmental Needs	I5T2

Test Case Identifier	I14T2
Test Case Description	A TaxiDriver accept a request
Test Item(s)	MobileClient → TaxiDriver
Input Specification	The positive answer input
Output Specification	If the reply is successful, a positive notification is shown, if not an error notification is shown
Environmental Needs	I10T1

Test Case Identifier	I14T3
Test Case Description	A TaxiDriver refuse a request
Test Item(s)	MobileClient → TaxiDriver
Input Specification	The negative answer input
Output Specification	If the reply is successful, a positive notification is shown, if not an error notification is shown
Environmental Needs	I10T2

Test Case Identifier	I14T4
Test Case Description	A TaxiDriver set his status as available
Test Item(s)	MobileClient → TaxiDriver
Input Specification	Nothing
Output Specification	The current status of the TaxiDriver is now shown as Available
Environmental Needs	I11T2

Test Case Identifier	I14T5
Test Case Description	A TaxiDriver set his status as unavailable
Test Item(s)	MobileClient → TaxiDriver
Input Specification	Nothing
Output Specification	The current status of the TaxiDriver is now shown as Unavailable
Environmental Needs	I11T1

3.1.15 Integration test case I15

Test Case Identifier	I15T1
Test Case Description	MobileClient performs a random input
Test Item(s)	MobileClient → System
Input Specification	Fill one of the available form of a MobileClient
Output Specification	Verify that the input form is correctly received by the system
Environmental Needs	All the server functionalities tested

Test Case Identifier	I15T2
Test Case Description	Send a random notification to the MobileClient
Test Item(s)	System → MobileClient
Input Specification	Send a test notification to a MobileClient
Output Specification	Verify that the notification is shown correctly
Environmental Needs	All the server functionalities tested

3.1.16 Integration test case I16

Test Case Identifier	I16T1
Test Case Description	WebClient performs a random input
Test Item(s)	WebClient → System
Input Specification	Fill one of the available form of a WebClient
Output Specification	Verify that the input form is correctly received by the system
Environmental Needs	All the server functionalities tested

Test Case Identifier	I16T2
Test Case Description	Send a random notification to the WebClient
Test Item(s)	System → WebClient
Input Specification	Send a test notification to a WebClient
Output Specification	Verify that the notification is shown correctly
Environmental Needs	All the server functionalities tested

4 Tools and Test Equipment Required

4.1 Arquillian

'The mission of the Arquillian project is to provide a simple test harness that developers can use to produce a broad range of integration tests for their Java applications...' Arquillian, a testing framework developed at JBoss.org, em-

powers the developer to write integration tests for business objects that are executed inside a container or that interact with the container as a client. The container may be an embedded or remote Servlet container, Java EE application server, Java SE CDI environment or any other container implementation provided. Arquillian strives to make integration testing no more complicated than basic unit testing.

It's obvious that this solution represents a good choice mainly if JEE is the environment chosen to develop our project, as we suggested in Design Document, otherwise it could be not compatible with the code written.

4.2 Manual testing

The manual testing of the interactions is always available. It may be not a wise choice, because there are no aids nor automations, but it could be the only way to test some parts of your code. The manual testing could be also slow and unsafe (paradox: it could be bugged itself), so we do not recommend to use it as a universal choice, but it could be a great solution if used in combination with a software tool to cover what the tool can not manage easily.

5 Program Stubs and Test Data Required

As we already said in section 2, we used a bottom-up approach with our testing strategy, therefore we utilized drivers instead of stubs.

5.1 Request Driver

The Request Driver is needed to test the correct functioning of the queue. In particular this driver is used to trigger the retrieval of the first taxi in the queue to whom the dummy request should be submitted. The dummy request will be made ad-hoc to trigger the method on the desired queue.

5.2 Guest Driver

The Guest Driver is needed to test the correct functioning of the Request interface. In particular this driver is used to create a request by a dummy Guest, that will insert random data to see if the request is generated correctly.

5.3 RegisteredPassenger Driver

The RegisteredPassenger Driver is needed to test the correct function of the Request interface. In particular this driver is used to create a Request by a dummy RegisteredPassenger, to see if the request is generated correctly. Also it is indirectly used during the test of the reservation since a reservation triggers a request once its time comes.

5.4 TaxiDriver Driver

The TaxiDriver Driver is needed to test the correct function of the TaxiRequest interface, and also to test the correct functioning of the queue. In particular this driver is used to give a positive answer or a negative one by a dummy TaxiDriver to a given request: in case of negative answer the TaxiDriver has to be put again in queue in the last position, while in case of positive answer the the system will ensure that the TaxiDriver is no longer in queue. The other use of this driver is to send an input to the system using the method SetAvailable or SetUnavailable. In the first case the TaxiDriver will be added in queue while in the second one the TaxiDriver will be removed from the queue.

6 Other Info

6.1 Hours of work

For each component follows an approximate indication of how much time was spent on the realization of this document

Component	Hours
Bernardis Cesare	16
Dagrada Mattia	14

6.2 Tools

We used various tools to develop this document:

- LaTeX 2_ε and TeXMaker editor 2.10.4
- Draw.IO