INTEGRATION TEST PLAN DOCUMENT

MYTAXISERVICE

Reference Professor: Elisabetta Di Nitto

POLITECNICO DI MILANO
Computer Science and Engineering
Project of Software Engineering 2

Andrea Binelli
858235

Summary

1. Introduction

1.1 Revision History

Document Title	Integration Test Plan Document
Document Identification	Deliveries/ITPD_v1.pdf
Document Version	1.0
Date Of Creation	21/01/2016
Changes	Creation of the document

1.2 Purpose and Scope

1.2.1 Purpose

The purpose of this document is to design a test plan for the integration of the components.

This document contains the description of the integration tests for the project myTaxService and it is one of five assignments for the course Software Engineering 2 at Politecnico di Milano.

1.2.2 Scope

The software to be integrated and tested is all the components of the application myTaxiService, described and formalized in the RASD and DD documents.

1.3 List of Definitions and Abbreviations

1.3.1 Definitions

Component	A software unit, in this document it can be one of this types: Main Tab, Entity, Manager
Driver	The program that accepts test data and passes this test to the component to be tested and prints relevant results.
Entity	The data model of the application.
Main Tab	It represents a complete tab (or a tab inside a web page) of our application. They are client side, they do not implement business logic and data (except for the coordinates of the taxi).
Manager	It Manage all the business logic of the system and act as a connection between the Main Tabs and the entities. They are server side.
Sub-System	A division of the system by aggregating related functions.

1.3.2 Abbreviations

DD	Requirement Analysis and Specification Document
ITPD	Integration Test Plan Document
RASD	Design Document
SIS	Software Integration Sequence
TP	Test Procedure

1.4 List of Reference Documents

- RASD, Requirement Analysis and Specification Document, myTaxiService, 2015 Version 2
- DD, Design Document, myTaxiService, 2015 Version 2

2. Integration Strategy

2.1 Entry Criteria

The criteria that must be met before integration testing of specific elements begin are descripted in the chapter 2.3.2.

Also, every test case must describe what the criteria are to pass that specific test.

2.2 Elements to be Integrated

In the DD document I have identified the subsystems (chapter 1.3) by functionality so many (software) elements can belong to different subsystems (for example the Guest Main Tab is visible both in the Sign Up an in the Log In).

I decide to group those subsystems in two bigger ones in order to reduce repetitions of components.

For each of two subsystems I identify the software components.

The elements can be parts of the graphical user (or taxi) interface (like tabs), software components like controllers (business logic) and entities (data model of the application).

2.2.1 Sub-System 1

Functions: Sign Up, Log In and Change Profile Data

Components: Guest Main Tab, User Main Tab, Profile Data Manager, Login Manager, User Entity

2.2.2 Sub-System 2

Functions: Taxi Managing, Request Managing, Request Log

Components: User Main Tab, Taxi Main Tab, Request Manager, Address Entity, Taxi Entity, Area Entity, Request Entity

2.3 Integration Testing Strategy

2.3.1 Testing Strategy

The type of test is black block because the code has to be done, I only specify high level methods in the Design Document (chapter 4)

For testing I choose the bottom-up approach. This means that integration testing starts at the bottom level. This way the project will be built up from the bottom level. The integration tests described in this documents are at the component level.

I will integrate first the Main Tabs with the managers and test all the methods assuming that I have already tested the singular components (in the unit tests). When I integrate them, in order to perform specific functions (like login function) I require a driver that simulates the entities (not yet integrated) that return sample data (like a password to be compared).

Then I will substitute the driver whit the entities and test the methods between the business logic and the data entities.

2.3.2 Testing Tasks

The following tasks are necessary for performing a specific integration test:

- Designing the integration test
- Designing a driver (if it was not made at the unit test)
- Designing input test data (if it was not made at the unit test)
- Setting up a system, the components involved, the driver and the input test data
- Performing the integration test

The following items must be delivered when a specific integration test is finished:

- Integration test report, which will be listed in the integration tests paragraph
- Integration test output data
- Problem reports (if necessary)

The following items must be delivered when testing on all code modules has finished:

- All integration test reports.

2.4 Sequence of Component/Function Integration

2.4.1 Components Integration Sequence

For each subsystem I identify the sequence in which the components will be integrated within the subsystem.

Sub-System 1:

Software Integration Sequences

SIS1: Guest Main Tab, User Main Tab -> Profile Data Manager

SIS2: Guest Main Tab, User Main Tab -> Login Manager

SIS3: Profile Data Manager, Login Manager -> User Entity

Sub-System 2:

Software Integration Sequences

SIS4: Taxi Main Tab -> Taxi Manager

SIS5: Taxi Main Tab, User Main Tab -> Request Manager

SIS6: Taxi Manager, Request Manager ->Taxi Entity

SIS7: Taxi Manager, Request Manager -> Area Entity

SIS8: Request Manager -> Address Entity

SIS9: Request Manager -> Request Entity

2.4.2 Subsystem Integration Sequence

The two subsystem are not strictly related but I find more appropriate to first integrate the subsystem 1 and then the subsystem 2 (because the User Main Tab it is accessible only after authentication).

3. Individual Steps and Test Description

3.1 Sub-System 1

3.1.1 Software Integration Sequence SIS1

Test Case Identifier	SIS1T1
Test Item(s)	Guest Main Tab -> Profile Data Manager
Input Specification	Create typical Guest Main Tab input
Output Specification	Check if the correct methods are called in the Profile Data Manager and the proper exceptions are caught
Environmental Needs	User Entity Driver

Test Case Identifier	SIS1T2
Test Item(s)	User Main Tab -> Profile Data Manager
Input Specification	Create typical User Main Tab input
Output Specification	Check if the correct methods are called in the Profile Data Manager and the proper exceptions are caught
Environmental Needs	User Entity Driver

3.1.2 Software Integration Sequence SIS2

Test Case Identifier	SIS2T1
Test Item(s)	Guest Main Tab -> Login Manager
Input Specification	Create typical Guest Main Tab input
Output Specification	Check if the correct methods are called in the Login Manager and the proper exceptions are caught
Environmental Needs	User Entity Driver

3.1.3 Software Integration Sequence SIS3

Test Case Identifier	SIS3T1
Test Item(s)	Profile Data Manager -> User Entity
Input Specification	Create typical Profile Data Manager input
Output Specification	Check if the correct methods are called in the User Entity
Environmental Needs	SIS1 succeeded

Test Case Identifier	SIS3T2
Test Item(s)	Login Manager -> User Entity
Input Specification	Create typical Login Manager input
Output Specification	Check if the correct methods are called in the User Entity
Environmental Needs	SIS2 succeeded

3.2 Sub-System 2

3.2.1 Software Integration Sequence SIS4

Test Case Identifier	SIS4T1
Test Item(s)	Taxi Main Tab -> Taxi Manager
Input Specification	Create typical Taxi Main Tab input
Output Specification	Check if the correct methods are called in the Taxi Manager
Environmental Needs	Taxi Entity, Area Entity Drivers

3.2.2 Software Integration Sequence SIS5

Test Case Identifier	SIS5T1
Test Item(s)	Taxi Main Tab -> Request Manager
Input Specification	Create typical Taxi Main Tab input
Output Specification	Check if the correct methods are called in the Request Manager
Environmental Needs	Taxi Entity, Area Entity, Address Entity Drivers

Test Case Identifier	SIS5T2
Test Item(s)	User Main Tab -> Request Manager
Input Specification	Create typical User Main Tab input
Output Specification	Check if the correct methods are called in the Request Manager and the proper exceptions are caught
Environmental Needs	Taxi Entity, Area Entity, Address Entity, Request Entity Drivers

3.2.3 Software Integration Sequence SIS6

Test Case Identifier	SIS6T1
Test Item(s)	Taxi Manager -> Taxi Entity
Input Specification	Create typical Taxi Manager input
Output Specification	Check if the correct methods are called in the Taxi Entity
Environmental Needs	SIS4 Succeeded

Test Case Identifier	SIS6T2
Test Item(s)	Request Manager-> Taxi Entity
Input Specification	Create typical Request Manager input
Output Specification	Check if the correct methods are called in the Taxi Entity
Environmental Needs	SIS5 Succeeded

3.2.4 Software Integration Sequence SIS7

Test Case Identifier	SIS7T1
Test Item(s)	Taxi Manager -> Area Entity
Input Specification	Create typical Taxi Manager input
Output Specification	Check if the correct methods are called in the Area Entity
Environmental Needs	SIS4 Succeeded

Test Case Identifier	SIS7T2
Test Item(s)	Request Manager-> Area Entity
Input Specification	Create typical Request Manager input
Output Specification	Check if the correct methods are called in the Area Entity
Environmental Needs	SIS5 Succeeded

3.2.5 Software Integration Sequence SIS8

Test Case Identifier	SIS8T1
Test Item(s)	Request Manager-> Address Entity
Input Specification	Create typical Request Manager input
Output Specification	Check if the correct methods are called in the Address Entity
Environmental Needs	SIS5 Succeeded

3.2.6 Software Integration Sequence SIS9

Test Case Identifier	SIS9T1
Test Item(s)	Request Manager-> Request Entity
Input Specification	Create typical Request Manager input
Output Specification	Check if the correct methods are called in the Request Entity
Environmental Needs	SIS5 Succeeded

3.3 Test Procedures

3.3.1 Test Procedure T1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the subsystem 1: • Can handle the Sign Up function • Can handle the Log In function • Can handle the Change Profile Data function • Can catch a “wrong login input” exception and show an error message • Can catch a “user not found” exception and show an error message • Can catch a “wrong authentication” exception and show an error message • Can catch a “bad profile data” exception and show an error message • Can catch a “email already exists” exception and show an error message • Can catch a “email or phone number already exists” exception and show an error message
Procedure Steps	Execute SIS3 after SIS1,SIS2

3.3.2 Test Procedure T2

Test Procedure Identifier	TP2
Purpose	This test procedure verifies whether the subsystem 2: • Can handle the Taxi Initialize function • Can handle the Taxi Availability ON/OFF functions • Can handle the Manage Taxi Position function • Can handle the Manage Request function • Can handle the Request Log function • Can catch a “bad address” exception and show an error message • Can catch a “no taxi” exception and show an error message
Procedure Steps	Execute SIS6, SIS7, SIS8, SIS9 after SIS4, SIS5

4. Tools and Test Equipment Required

Manual Testing are required for testing the functional requirements (descripted in the RASD) during the integration.

For the non-functional requirements (performance and reliability) I need a specific tool that simulate a load on the server.

For this web application I can user JMeter that is A GUI desktop application designed to load test functional behaviour and measure performance. It can be used to simulate a heavy load on a server, network or object to test its strength or to analyse overall performance under different load types.

I need to use it for the server side, because there is no need to test performance on the client side.

5. Program Stubs and Test Data Required

For each software integration test identified in the chapter 3 I explain better the data and methods to be tested and the expected behaviours in order to fulfil the purpose of the test procedures:

5.1 SIS1

Methods called in the Profile Data Manager to be tested: verifyProfileData()

Data to be tested	Expected behaviour
The email inserted already exists	“email already exists” exception and an error message
The phone number inserted already exists (if it is inserted)	“email or phone number already exists” exception and show an error message
Some required field not filled	“bad profile data” exception and an error message
Password is shorter than 8 chars	“bad profile data” exception and an error message
The confirming password is not matching the password	“bad profile data” exception and an error message
Password is shorter than 8 chars	“bad profile data” exception and an error message
Old password does not match with the current password (only in the Change Profile Data function)	“bad profile data” exception and an error message
Correct input	SignUp/Change Profile Data function accomplished and Main Tab showed

5.2 SIS2

Methods called in the Login Manager to be tested: checkLoginFields()

Data to be tested	Expected behaviour
The email inserted is wrong (it doesn’t match with any yet registered emails)	“user not found” exception and an error message
The phone number inserted is wrong (it doesn’t match with any yet registered phone number)	“user not found” exception and show an error message
Some required field not filled	“wrong login input” exception and an error message
The password inserted is not matching the one associated with the account	“wrong authentication” exception and an error message
Correct input	SignUp/Change Profile Data function accomplished and Main Tab showed

5.3 SIS3

Methods called in the User Entity to be tested: findByEmail(), findByPhoneNumber()

Data to be tested	Expected behaviour
Email doesn’t match with any yet registered emails	“user not found” exception
Phone number doesn’t match with any yet registered phone number	“user not found” exception
Correct email/phone number	Return user data

5.4 SIS4:

Methods called in the Taxi Manager to be tested: loadTaxiCode(), manageAvailability()

Data to be tested	Expected behaviour
Taxi out of the city turn on availability	The switch is turned off by the manager
Taxi in the city turn on availability	Nothing
Available taxi change zone	Nothing
Available taxi goes out the city	The switch is turned off by the manager

5.5 SIS5:

Methods called in the Request Manager to be tested: requestTaxi(), sendRequest(), sendResponse(), createRequest(), updateRequest(), deleteRequest()

Data to be tested	Expected behaviour
Call from User Main Tab with an inexistent address	"bad address" exception and an error message to User Main Tab
Call from User Main Tab with no taxi available in that zone	"no taxi" exception and an error message to User Main Tab
Call from User Main Tab with every (available) taxis clicks no and timeout occurs	"no taxi" exception and an error message to User Main Tab
Call from User Main Tab and a taxi clicks yes	The switch of the Taxi is turned off by the manager and a confirm is sent to the User Main Tab

5.6 SIS6:

Methods called in the Taxi Entity to be tested: getCode(), getArea(), setArea()

No exceptions are expected

5.7 SIS7:

Methods called in the Area Entity to be tested: getLatitudeTop(), getLatitudeBottom(), getLongitudeLeft, getLongitudeRight(), searchByArea(), removeTaxi(), addTaxi(), getAdjAreas(), getFirstTaxi(), moveFromFirstToLast()

No exceptions are expected

5.8 SIS8:

Methods called in the Address Entity to be tested: getAddress(), getArea()

Data to be tested	Expected behaviour
Address name doesn't match with any yet registered address	"bad address" exception
Correct address name	Return data

5.9 SIS9:

Methods called in the Request Entity to be tested: setAddress(), setUser(), setState(), setWaitTime(), setTaxi()

No exceptions are expected

For redacting and writing this document I spent about 15 hours.