



MYTAXYSERVICE®

## INTEGRATION TEST PLAN

MATTEO GRESELIN matr. 773600 ,  
SEYEED MOHSEN KASHFI HAGHIGHI matr.859085

Version 1.1

# Contents

<b>1.Introduction</b>	<b>2</b>
1.1 Revision History . . . . .	2
1.2 Purpose and Scope . . . . .	2
1.3 Definition and abbreviation . . . . .	2
1.4 Reference . . . . .	3
<b>2. Integration Strategy</b>	<b>4</b>
2.1 Entry Criteria . . . . .	4
2.2 Elements to be Integrated . . . . .	4
2.3 Integration Testing Strategy . . . . .	5
2.4 Sequence of components And Function Integration . . . . .	5
2.4.1 Software And Subsystem Integration Sequence . . . . .	7
<b>3. Individual Steps and Test Description</b>	<b>8</b>
3.1 Integration Test Case I1 . . . . .	8
3.2 Integration Test Case I2 . . . . .	8
3.3 Integration Test Case I3 . . . . .	9
3.4 Integration Test Case I4 . . . . .	9
3.5 Integration Test Case I5 . . . . .	10
3.6 Integration Test Procedure TP1 . . . . .	10
3.7 Integration Test Procedure TP2 . . . . .	11
3.8 Integration Test Procedure TP3 . . . . .	11
3.9 Integration Test Procedure TP4 . . . . .	11
<b>4. Tools and Test Equipements Required</b>	<b>12</b>

# 1. Introduction

## 1.1 Revision History

Date	Version	Description	Authors
17/01/2016	0.1	Create draft of document	Mohsen Kashfi, Matteo Greselin
19/01/2016	1.0	First version prepared for the deadline. Future revision could be done for the final presentation	Mohsen Kashfi, Matteo Greselin
21/01/2016	1.1	Last changes before upload document on GitHub	Mohsen Kashfi, Matteo Greselin

## 1.2 Purpose and Scope

Testing is the process of evaluating a system and verify that it satisfies specified requirement. In this document we are going to Reduce the ambiguity of the system, in the other words our purpose is to enhance the quality of the product. Quality improvements help the organization to reduce support and service expenses. Moreover providing a good quality product causes the confidence of the customers. Some issues should be done in order to have a good quality system. For example: the usability of the system, reducing the defects/bugs, it is very important to verify and validate the product to meets the stated requirement/specification. Verification and Validation are performed to ensure that we are building the right system. Overall our main purpose is, to design tests that can specify differentiate types of defects/errors with the least time and price and ensure that unit tested modules interact correctly.

## 1.3 Definition and Abbreviation

- RASD: Requirement and specification document
- DD: Design Document
- Client: Application that is used by all the users except the resource provider who uses the agent application.
- ITP: Integration Test Plan
- CSM: Client Side Manager

- R&R: Request\Reservation
- GFM: General functionalities manager

## 1.4 Reference

For draft Integration Test Plan Document we have used the following items:

- Project Description
- Assignments provided by professor
- Slide of lecture
- RASD
- DD
- Internet and Software Engineering Forum on Beep for clarification

## 2. Integration Strategy

### 2.1 Entry Criteria

The following points must be verified before integration testing could begin:

1. All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system including (RASD-DD).
2. All test hardware platforms must have been successfully installed, configured and functioning properly.
3. Unit testing of the individual services should be conducted. Our individual services are divided into subsystems by functions.
4. The correctness of the database should be tested
5. Source code must be unit tested and available
6. The test environment such as, lab, hardware, software, and system administration support should be ready

### 2.2 Elements to be Integrated

With respect to our design document, our subsystem are: TaxiDriver, Passenger ,Administrator , PoliciesAndData, ClientSideManager and GeneralFunctionalities.

Moreover each components have its own individual system for instance,

- Taxi driver:
  - Accept\Refuse,
  - Availability\Unavailability
  - Other specific functions.
- Passenger:
  - R&R Some
  - extra functions.
- Administrator:

- Administrator functionalities such as: Reporting, Sending any notification and system observation.
- GeneralFunctionalities:
  - Profile Managing,
  - helps
  - System situation.
- PoliciesAndDatas:
  - permanent dates,
  - volatile dates
  - Main policies.
- ClientManager:
  - Sign up,
  - Login
  - Visitors (as Guests)

## 2.3 Integration Testing Strategy

We have decided to use a functional grouping approach. This choice has been done because there are subsystems that require that other one before them working correctly.

For instance : if there is an error in the sign up process, log in process could never be correct because it never found nothing in the database also if user insert a correct couple of datas (this means a correct corresponding between username and password).

With respect to this idea we have preferred to approach testis as a user : before like guest, then sign up, log in.....

After each test we expect that system will create a report that will describe how the test is gone and if it was successfully concluded.

## 2.4 Sequence of components And Function Integration

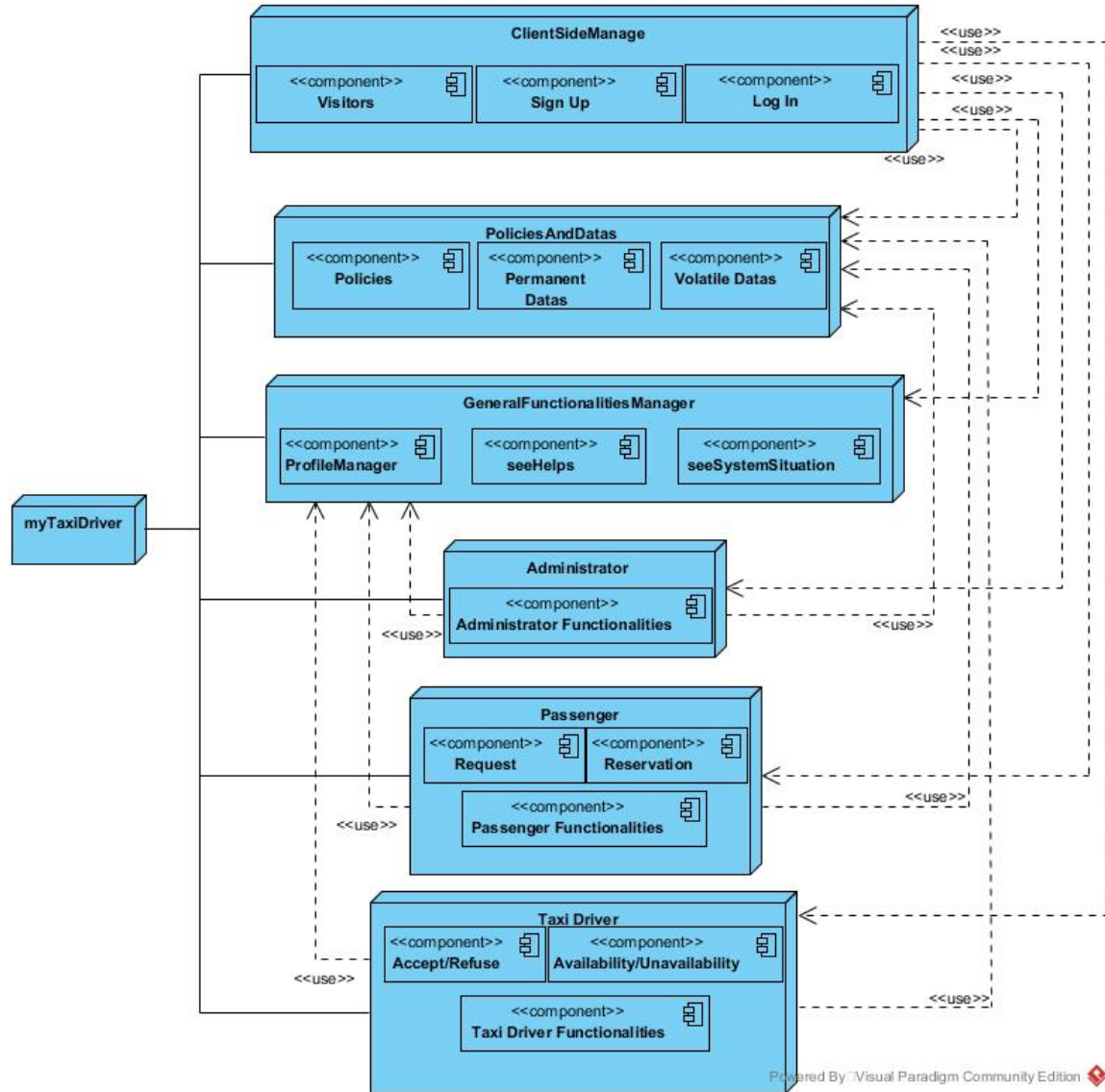
- Visitor
- Sign Up
- Log in

As far as there are several types of logging (Passenger, Taxi Driver or Administrator) we must consider that, Passenger component is in priority and must be tested before taxi driver, then for the administrator, no matter is going to be done. According to our constraints we should have the following order:

- R&R
- Passenger functions
- Availability/Unavailability
- Accept/Refuse
- Taxi Functions
- Admin functions

## 2.4.1 Software And Subsystem Integration Sequence

As we have exposed in our DD, next image represents the division in subsystem of our system.



ID	Integration Test	Paragraphs
I1	ClientSideManager GeneralFunctionalitiesManager , PoliciesAndDatas	3.1 - 3.6
I2	ClientSideManager Passenger , TaxiDriver, Administrator	3.2 - 3.7 - 3.8 - 3.9
I3	Passenger PoliciesAndDatas , GeneralFunctionalitiesManager	3.3 - 3.7
I4	TaxiDriver PoliciesAndDatas , GeneralFunctionalitiesManager	3.4 - 3.8
I5	Administrator PoliciesAndDatas , GeneralFunctionalitiesManager	3.5 - 3.9



## 3 Individual Steps and Test Description

### 3.1 Integration Test Case I1

<b>TEST CASE ID</b>	I1T1
<b>TEST ITEMS</b>	ClientSideManager GeneralFunctionalitiesManager
<b>INPUT SPECIFICATION</b>	Create typical ClientSideManager input
<b>OUTPUTSPECIFICATION</b>	Check the correctness of the functions which is called in the GFM
<b>ENVIROMENTAL NEEDS</b>	Connection between Client and Server is required
<b>TEST CASE ID</b>	I1T2
<b>TEST ITEMS</b>	ClientSideManager PoliciesAndDatas
<b>INPUT SPECIFICATION</b>	Create typical ClientSideManager input
<b>OUTPUTSPECIFICATION</b>	Check the correctness of the function which is called in policies and datas
<b>ENVIROMENTAL NEEDS</b>	Connection between Client and Server is required

### 3.2 Integration Test Case I2

<b>TEST CASE ID</b>	I2T1
<b>TEST ITEMS</b>	ClientSideManager Passenger
<b>INPUT SPECIFICATION</b>	Create typical ClientSideManager input
<b>OUTPUTSPECIFICATION</b>	Check the correctness of the function which is called in Passenger
<b>ENVIROMENTAL NEEDS</b>	I1 succeeded

TEST CASE ID	I2T2
TEST ITEMS	ClientSideManager TaxiDriver
INPUT SPECIFICATION	Create typical CSM input
OUTPUTSPECIFICATION	Check the correctness of the function which is called in TD
ENVIROMENTAL NEEDS	I1 succeeded

TEST CASE ID	I2T3
TEST ITEMS	ClientSideManager Administrator
INPUT SPECIFICATION	Create typical ClientSideManager input
OUTPUTSPECIFICATION	Check the correctness of the function which is called in Administrator
ENVIROMENTAL NEEDS	I1 succeeded

### 3.3 Integration Test Case I3

TEST CASE ID	I3T1
TEST ITEMS	Passenger PoliciesAndDatas
INPUT SPECIFICATION	Create typical Passenger input
OUTPUTSPECIFICATION	Check the correctness of the function which is called in PoliciesAndDatas
ENVIROMENTAL NEEDS	I1, I2T1 succeeded

TEST CASE ID	I3T2
TEST ITEMS	Passenger GeneralFunctionalitiesManager
INPUT SPECIFICATION	Create typical Passenger input
OUTPUTSPECIFICATION	Check the correctness of the functions which is called in the GFM
ENVIROMENTAL NEEDS	I1, I2T1 succeeded

### 3.4 Integration Test Case I4

TEST CASE ID	I4T1
TEST ITEMS	TaxiDriver PoliciesAndDatas
INPUT SPECIFICATION	Create typical TD input
OUTPUTSPECIFICATION	Check the correctness of the function which is called in PoliciesAndDatas
ENVIROMENTAL NEEDS	I1, I2T2 succeeded

<b>TEST CASE ID</b>	<b>I4T2</b>
<b>TEST ITEMS</b>	<b>TaxiDriver PoliciesAndDatas</b>
<b>INPUT SPECIFICATION</b>	<b>Create typical TD input</b>
<b>OUTPUTSPECIFICATION</b>	<b>Check the correctness of the function which is called in PoliciesAndDatas</b>
<b>ENVIROMENTAL NEEDS</b>	<b>I1, I2T2 succeeded</b>

<b>TEST CASE ID</b>	<b>I4T3</b>
<b>TEST ITEMS</b>	<b>TaxiDriver GeneralFunctionalitiesManager</b>
<b>INPUT SPECIFICATION</b>	<b>Create typical TD input</b>
<b>OUTPUTSPECIFICATION</b>	<b>Check the correctness of the functions which is called in the GFM</b>
<b>ENVIROMENTAL NEEDS</b>	<b>I1, I2T1 succeeded</b>

### 3.5 Integration Test Case I5

<b>TEST CASE ID</b>	<b>I5T1</b>
<b>TEST ITEMS</b>	<b>Administrator PoliciesAndDatas</b>
<b>INPUT SPECIFICATION</b>	<b>Create typical Admin input</b>
<b>OUTPUTSPECIFICATION</b>	<b>Check the correctness of the function which is called in PoliciesAndDatas</b>
<b>ENVIROMENTAL NEEDS</b>	<b>I1, I2T3 succeeded</b>

<b>TEST CASE ID</b>	<b>I5T2</b>
<b>TEST ITEMS</b>	<b>Administrator GeneralFunctionalitiesManager</b>
<b>INPUT SPECIFICATION</b>	<b>Create typical Admin input</b>
<b>OUTPUT SPECIFICATION</b>	<b>Check the correctness of the functions which is called in the GFM</b>
<b>ENVIROMENTAL NEEDS</b>	<b>I1, I2T1 succeeded</b>

### 3.6 Integration Test Procedure TP1

<b>TEST PROCEDURE ID</b>	<b>TP1</b>
<b>PURPOSE</b>	<p>This test procedure verify that client software, in precise condition:</p> <ul style="list-style-type: none"> <li>- can handle public pages and informations</li> <li>- can output required information to the client</li> <li>- can not modify personal info</li> <li>- can not require or reserve any Taxi</li> <li>- can not accept or refuse any request or reservation</li> <li>- can not set availability</li> </ul>
<b>PROCEDURE STEPS</b>	<b>Execute I1</b>

### 3.7 Integration Test Procedure TP2

TEST PROCEDURE ID	TP2
PURPOSE	This test procedure verify that client software, in precise condition: <ul style="list-style-type: none"><li>- can handle public pages and informations</li><li>- can modify personal info concistently with the log in done</li><li>- can require only one Taxi</li><li>- can reserve only one Taxi</li><li>- can output required information to the client</li><li>- can not accept or refuse any request or reservation</li><li>- can not set availability</li></ul>
PROCEDURE STEPS	Execute I2T1 and I3 after I1

### 3.8 Integration Test Procedure TP3

TEST PROCEDURE ID	TP3
PURPOSE	This test procedure verify that client software, in precise condition: <ul style="list-style-type: none"><li>- can handle public pages and informations</li><li>- can modify personal info concistently with the log in done</li><li>- can accept or refuse request or reservation</li><li>- can set availability</li><li>- can output required information to the client</li><li>- can not require or reserve any Taxi</li></ul>
PROCEDURE STEPS	I2T2 and I4 after I1

### 3.9 Integration Test Procedure TP4

TEST PROCEDURE ID	TP4
PURPOSE	This test procedure verify that client software, in precise condition: <ul style="list-style-type: none"><li>- can handle public pages and informations</li><li>- can modify personal info concistently with the log in done</li><li>- can delete users</li><li>- can send personal messages to users</li><li>- can output required information to the client</li><li>- can not require or reserve any Taxi</li><li>- can not accept or refuse any request or reservation</li><li>- can not set availability</li></ul>
PROCEDURE STEPS	Execute I2T3 and I5 after I1

## 4. Tools and Test Equipements Required

MyTaxiService will be an application with a very variable load on servers. For instance these days with the strike of public transportation, we must provide a heavy load on Passenger Server. According to this idea we could use JMeter for testing stress situation for our system in order to avoid any possibilities that it should go out of work.

Another problem that we must consider is that, when we will be sure that each subsystem works coorectly, we will not be sure that they will communicate correctly eachother. To solve this possible situation could be usefull testing communication with Arquillian, a tool that provides methods for checking the correctness of interaction between elements.

