



Politecnico di Milano, A.A. 2015/2016

Software Engineering 2: My Taxi Service  
**Integration Testing Plan Document**

Belotti Nicola 793419  
Chioso Emanuele 791621  
Colombo Andrea 853381

January 20, 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Revision History . . . . .	2
1.2	Purpose . . . . .	2
1.3	List of Reference Documents . . . . .	2
<b>2</b>	<b>Integration Strategy</b>	<b>3</b>
2.1	Entry Criteria . . . . .	3
2.2	Elements to be Integrated . . . . .	3
2.3	Integration Testing Strategy . . . . .	3
2.4	Sequence of Component . . . . .	3
<b>3</b>	<b>Individual Steps and Test Description</b>	<b>5</b>
3.1	Integration test cases . . . . .	5
3.1.1	Integration test case I1 - REGISTRATION . . . . .	5
3.1.2	Integration test case I2 - LOGIN . . . . .	5
3.1.3	Integration test case I3 - RIDE CREATION . . . . .	5
3.1.4	Integration test case I4 - REQUEST . . . . .	6
3.1.5	Integration test case I5 - RESERVATION . . . . .	6
3.1.6	Integration test case I6 - JOIN . . . . .	6
3.1.7	Integration test case I7 - UNJOIN . . . . .	7
3.1.8	Integration test case I8 - CANCEL RESERVATION . . . . .	7
3.1.9	Integration test case I9 - DRIVER ACCEPTANCE/REFUSE . . . . .	7
3.1.10	Integration test case I10 - DRIVER STATUS CHANGE . . . . .	8
<b>4</b>	<b>Tools and Test Equipment Required</b>	<b>9</b>
<b>5</b>	<b>Program Stubs and Test Data Required</b>	<b>10</b>

# **1 Introduction**

## **1.1 Revision History**

- 15/01/2016 First redaction of the document

## **1.2 Purpose**

This document is the Integration Testing Plan Document for myTaxiService project. The purpose of this document is to list all the tests that will be performed on the my taxi service application. In particular we will focus on the integration part, describing how the test will be executed, which components will be tested and in which order. We will also list all the tools used to perform the integration tests.

## **1.3 List of Reference Documents**

- MyTaxiService Requirement Analysis and Specification document: `RASD.pdf`
- MyTaxiService Design document: `DD.pdf`
- Assignment 4: integration test plan: `Plan.pdf`
- Integration Test Plan Example document: Integration Test Plan: `Example.pdf`

## **2 Integration Strategy**

### **2.1 Entry Criteria**

Before integration tests may begin all the primary functions and components of the application must be finished and working. Specifically: registration, login, requests, reservation and taxi sharing functions must work as planned. To do so all the components listed in the design document must be working as well. Exception made for the user interface component.

### **2.2 Elements to be Integrated**

The components to be integrated are:

- Client component
- Ride manager component
- User manager component.

For a more detailed description on how these components should work refer to architectural design section of the Design Document

### **2.3 Integration Testing Strategy**

We will adopt a bottom-up testing strategy, integrating first the sub-components and later on the higher level. We choose this strategy because the sub-components of our system are independent one to each other and they can be integrated separately.

### **2.4 Sequence of Component**

Since we adopt a bottom-up integration strategy, we will start from the lower level component (database manager), then the components that directly access the database, thus account and rider creator, then the manager of those components and finally the user interface.

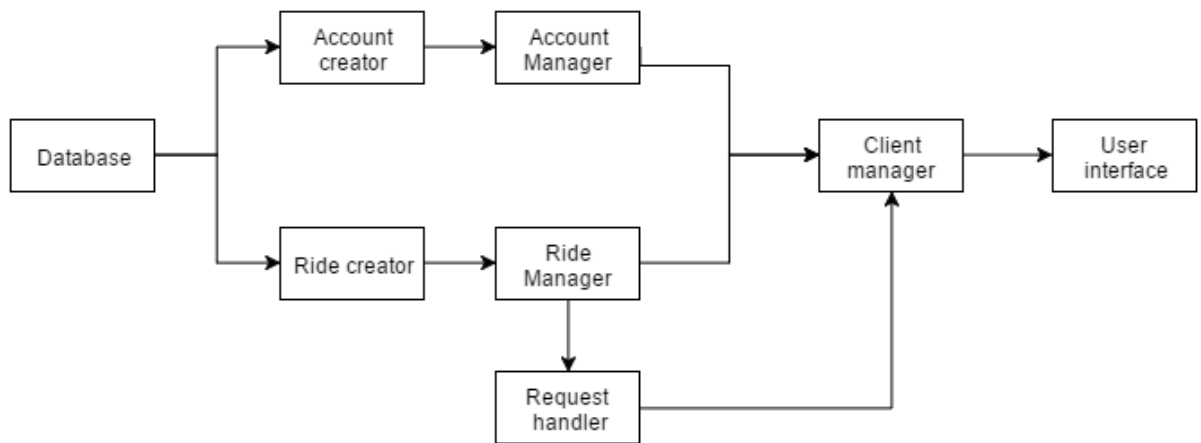


Figure 1: This flowchart show the sequence of integration of the components

## 3 Individual Steps and Test Description

### 3.1 Integration test cases

#### 3.1.1 Integration test case I1 - REGISTRATION

<b>Test Procedure Identifier</b>	I1T1
<b>Test Item(s)</b>	Account Creator → Database Manager
<b>Input Specification</b>	Create a typical and well formed Account Creator input.
<b>Output Specification</b>	Check if the Database Manager fulfills the tasks given by the Account Creator and if the correct methods are called in the Database Manager
<b>Description</b>	The test must check if every type of given methods work fine and if the Database Manager creates the account in the correct way with an INSERT.
<b>Environmental Needs</b>	Database available

#### 3.1.2 Integration test case I2 - LOGIN

<b>Test Procedure Identifier</b>	I2T1
<b>Test Item(s)</b>	Client Manager → Account Manager
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct functions are called properly in the Account Manager
<b>Environmental Needs</b>	I1 succeeded, Database working

#### 3.1.3 Integration test case I3 - RIDE CREATION

<b>Test Procedure Identifier</b>	I3T1
<b>Test Item(s)</b>	Ride Creator → Database Manager
<b>Input Specification</b>	Create a typical and well formed Ride Creator input.
<b>Output Specification</b>	Check if the correct methods are called in the Database Manager.
<b>Test Description</b>	
<b>Environmental Needs</b>	Database available

#### 3.1.4 Integration test case I4 - REQUEST

<b>Test Procedure Identifier</b>	I4T1
<b>Test Item(s)</b>	Client Manager → Request Handler
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Request Handler.
<b>Test Description</b>	
<b>Environmental Needs</b>	Database working

#### 3.1.5 Integration test case I5 - RESERVATION

<b>Test Procedure Identifier</b>	I5T1
<b>Test Item(s)</b>	Client Manager → Request Handler
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Request Handler.
<b>Test Description</b>	
<b>Environmental Needs</b>	I1, I3 succeeded

#### 3.1.6 Integration test case I6 - JOIN

<b>Test Procedure Identifier</b>	I6T1
<b>Test Item(s)</b>	Client Manager → Ride Manager
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Ride Manager.
<b>Test Description</b>	
<b>Environmental Needs</b>	I3, I4,I6 succeeded.

### 3.1.7 Integration test case I7 - UNJOIN

<b>Test Procedure Identifier</b>	I7T1
<b>Test Item(s)</b>	Client Manager → Ride Manager
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Ride Manager.
<b>Test Description</b>	
<b>Environmental Needs</b>	I3, I4, I6 succeeded.

### 3.1.8 Integration test case I8 - CANCEL RESERVATION

<b>Test Procedure Identifier</b>	I8T1
<b>Test Item(s)</b>	Client Manager → Ride Manager
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Ride Manager.
<b>Test Description</b>	Tests must check every typical situation with the normal reservations and the shared rides as explained in the RASD.pdf
<b>Environmental Needs</b>	I3 succeeded.

### 3.1.9 Integration test case I9 - DRIVER ACCEPTANCE/REFUSE

<b>Test Procedure Identifier</b>	I9T1
<b>Test Item(s)</b>	Client Manager → Ride Manager
<b>Input Specification</b>	Create a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Ride Manager.
<b>Test Description</b>	
<b>Environmental Needs</b>	I4 succeeded



### 3.1.10 Integration test case I10 - DRIVER STATUS CHANGE

<b>Test Procedure Identifier</b>	10T1
<b>Test Item(s)</b>	Client Manager → Account Manager
<b>Input Specification</b>	CCreate a typical and well formed Client Manager input.
<b>Output Specification</b>	Check if the correct methods are called in the Account Manager.
<b>Test Description</b>	
<b>Environmental Needs</b>	-

## 4 Tools and Test Equipment Required

The software tools used to automate the integration testing are the following:

**Apache JMeter** JMeter<sup>1</sup> is a powerful tool which may be used to test the performance of subsystems:

**Web tier:** simulate a heavy load on the web tier in order to check if the requirements on the maximum number of simultaneously connected users and on the response times stated in the RASD [?, p. 57] are respected. Performance testing on the web tier is described in ??.

**Business tier:** simulate a heavy load on the REST API. Please note that a stress test on the web tier as described before can also overload the business tier; tests on both sides are useful to identify the bottlenecks. Performance testing on the business tier is described in ??.

**JUnit** JUnit<sup>2</sup> is the most used framework for unit testing in Java. We plan to use it for unit tests of the single components (not covered by this document), but it is also used to do integration testing together with Mockito and Arquillian.

**Arquillian** Arquillian<sup>3</sup> is a test framework which can also manage the test of the containers and their integration with JavaBeans (dependency injection). We mainly use it for that purpose.

**Mockito** Mockito<sup>4</sup> is an open-source test framework useful to generate mock objects, stubs and drivers. We use it in several test cases to mock stubs and drivers for the components to test.

---

<sup>1</sup><http://jmeter.apache.org/>

<sup>2</sup><http://junit.org/>

<sup>3</sup><http://arquillian.org/>

<sup>4</sup><https://en.wikipedia.org/wiki/Mockito>

## 5 Program Stubs and Test Data Required