POLITECNICO
MILANO 1863

A.Y 2015-2016
Software Engineering 2: "myTaxiService"

Integration **T**est **P**lan **D**ocument

Version 1.0

Massimiliano Paci (mat. 852720)
Giovanni Patruno (mat. 852658)

January 21th 2016

Contents

1 Introduction **3**
- 1.1 Revision History . . . 3
- 1.2 Purpose . . . 3
- 1.3 Scope . . . 3
- 1.4 Definitions, Acronyms, Abbreviations . . . 3
  - 1.4.1 Acronyms . . . 3
  - 1.4.2 Abbrevations . . . 3
- 1.5 List of Reference Documents . . . 3

2 Integration Strategy **4**
- 2.1 Overview . . . 4
- 2.2 Entry Criteria . . . 4
- 2.3 Elements to be Integrated . . . 4
- 2.4 Integration Testing Strategy . . . 5
- 2.5 Sequence of Component/Function Integration . . . 5
  - 2.5.1 Software Integration Sequence . . . 5

3 Individual Steps and Test Description **6**
- 3.1 Test Case I0 . . . 6
- 3.2 Test Case I1 . . . 6
- 3.3 Test Case I2 . . . 6
- 3.4 Test Case I3 . . . 6
- 3.5 Test Case I4 . . . 6
- 3.6 Test Case I5 . . . 7
- 3.7 Test Case I6 . . . 7
- 3.8 Test Case I7 . . . 7
- 3.9 Test Case I8 . . . 7
- 3.10 Test Case I9 . . . 7
- 3.11 Test Case I10 . . . 7
- 3.12 Test Case I11 . . . 8
- 3.13 Test Case I12 . . . 8

4 Tools and Test Equipment Required **9**

5 Program Stubs and Test Data Required **9**

6 Appendix **10**
- 6.1 Software and tool used . . . 10
- 6.2 Working Hours . . . 10

1 Introduction

1.1 Revision History

21/01/01 : Version 1.0

1.2 Purpose

The purpose of the Integration Test Plan Document is to describe the integration test required for the components of myTaxiService application.

1.3 Scope

The aim of this project is to create a platform, named **myTaxiService**, to optimize the taxi service of a large city.

Passengers will be able to book a taxi for a certain route, view a cost preview and the relative waiting time.

Taxi drivers will be able to inform the system about their availability in a certain zone, confirm or deny a ride and consult the feedback of the passengers.

The city is divided in taxi zones and each one is associated to a queue of taxis. The system assigns every booking to an available driver in the ride's zone. If there is not a driver available in that specific zone, the system automatically assigns another available driver from another zone.

Taxi drivers will have access to the platform too: they will be able to check their queue state and and also see statistics on their routes.

The platform will be scalable and it could be integrated with other applications to implement new functionalities.

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Acronyms

- DD: Design Document
- RASD: Requirement Analysis and Specification Document

1.4.2 Abbrevations

- w.r.t. : with respect to
- [I-n]: n-integration test

1.5 List of Reference Documents

- Requirement Analysis and Specification Document [RASD]
- Design Document [DD]
- Project's assignments

2 Integration Strategy

2.1 Overview

In this section we will show the modules of our application and the interactions between them that have to be tested.

2.2 Entry Criteria

The following prerequisites have to be satisfied:

- Unit tests have been already performed
- The logic consistence of the system have been verified (see the alloy report in the RASD document)
- Interfaces to be tested must be already coded

2.3 Elements to be Integrated

We decided to design, with a different point of view respect to diagrams present in the DD and RASD (Component and Class diagrams), the modules of myTaxiService application in order to clarify with more accuracy the functionalities and their interactions.
We identified the following modules:

- Request Management
- Notification Management
- Booking Management
- User Management
- Payment Management
- Booking Creation / Modification
- Booking Cancellation
- Authentication
- Authorization
- Queue Management
- Creation / Modification Validation
- Deletion Validation
- Booking Insertion
- Booking Deletion

2.4 Integration Testing Strategy

We use a Top-Down approach for testing. This means that we start from the high level modules going to the lower ones.

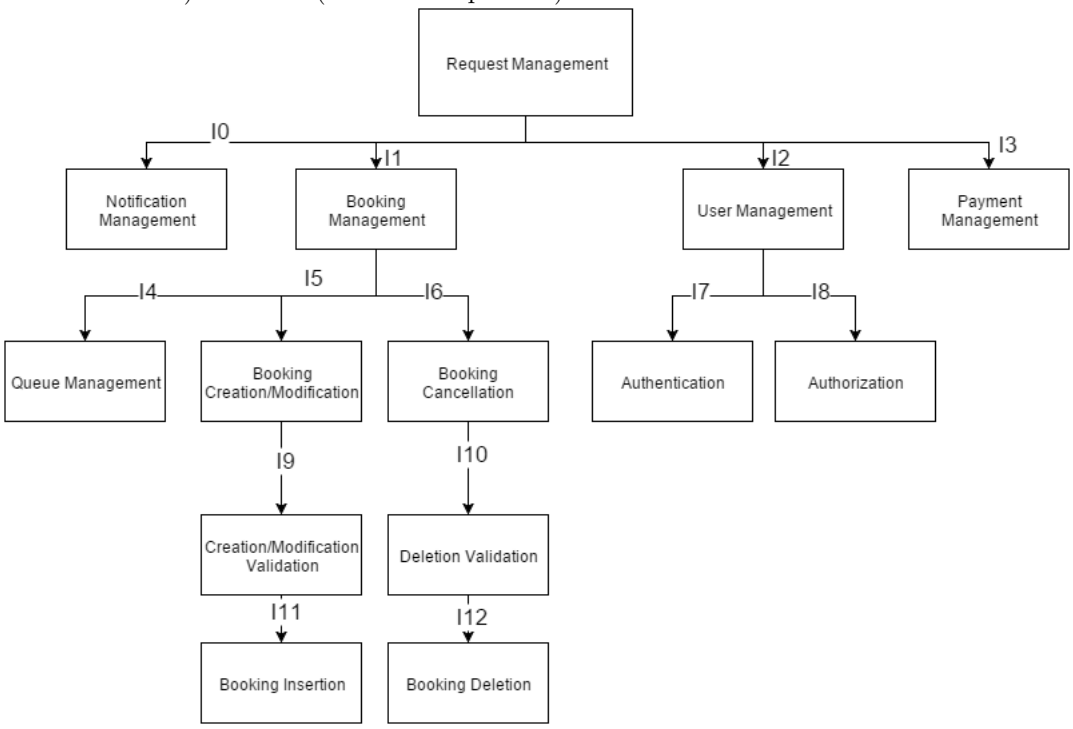
We chose a hierarchical approach (top-down or bottom-up) because fits better with the designed structure.

2.5 Sequence of Component/Function Integration

In this section we will present the integration sequence for the components of the system.

2.5.1 Software Integration Sequence

We assigned interfaces' label w.r.t. the order in which they will be tested. The following diagram shows, in a clear way, which are the integration test that we have to perform. Blocks represent the modules and edges the interactions between them. Each block represent an aggregation of subsystem mentioned in the design document (DD) that could be a multiple class integration (MVC) and a multiple integration between different subsystems, such as a guest (client or subscriber) or a host (server or dispatcher). We will test these interactions.



3 Individual Steps and Test Description

3.1 Test Case I0

Test item(s)	NotificationManagement ↔ RequestManagement
Input specification	Generate a single trigger event to notify a user or a set of users
Output specification	Checks if the dispatcher methods of the classes are called correctly
Environmental needs	Client stub, Event stub

3.2 Test Case I1

Test item(s)	BookingManagement ↔ RequestManagement
Input specification	Generate a request for a general booking management operation
Output specification	Checks if the correct methods for the requested operation are called
Environmental needs	Client stub

3.3 Test Case I2

Test item(s)	UserManagement ↔ RequestManagement
Input specification	Generate a request of a general guest operation (e.g. logging in)
Output specification	Checks if the correct methods for the request are called.
Environmental needs	Client stub

3.4 Test Case I3

Test item(s)	PaymentManagement ↔ RequestManagement
Input specification	Generate a request of a certain payment
Output specification	Checks if the responses of the API payment methods are consistent
Environmental needs	I2 Passed, Booking stub

3.5 Test Case I4

Test item(s)	BookingManagement ↔ QueueManagement
Input specification	Generate a trigger event (timeout on the waiting for the taxi-driver finding) for a specific booking
Output specification	Checks if the taxi driver is found and queue is managed correctly
Environmental needs	I1 Passed, GPS coordinates stub

3.6 Test Case I5

Test item(s)	BookingCreation-Modification ↔ BookingManagement
Input specification	Generate a booking creation or modification request
Output specification	Checks if the correct methods (creation or modifications) are invoked.
Environmental needs	I1 Passed

3.7 Test Case I6

Test item(s)	BookingCancellation ↔ BookingManagement
Input specification	Generate a booking cancellation request
Output specification	Checks if the correct deleting methods are called
Environmental needs	I1 Passed

3.8 Test Case I7

Test item(s)	Authenication ↔ UserManagement
Input specification	An authentication request is being processed
Output specification	Checks that the status of the user (authenticated or not) is consistent
Environmental needs	I2 Passed, Client stub

3.9 Test Case I8

Test item(s)	Authorization ↔ UserManagement
Input specification	An authorization request is being processed
Output specification	The permission's status of the user (admin, etc) is consistent
Environmental needs	I2 passed

3.10 Test Case I9

Test item(s)	BookingCreationModification ↔ CreationModificationValidation
Input specification	There is a specific request of validation
Output specification	The booking creation / modification is/ is not validated
Environmental needs	I4 passed

3.11 Test Case I10

Test item(s)	BookingCancellation ↔ DeletionValidation
Input specification	There is a specific request of validation
Output specification	The booking cancellation is /is not validated
Environmental needs	I5 passed

3.12 Test Case I11

Test item(s)	CreationModificationValidation ↔ BookingInsertion
Input specification	A validated creation /modification request is processed
Output specification	Booking added/modified in the database
Environmental needs	I8 passed

3.13 Test Case I12

Test item(s)	DeletionValidation ↔ BookingDeletion
Input specification	A validated deletion request is processed
Output specification	Bookinh deleted from the database
Environmental needs	I9 passed

4 Tools and Test Equipment Required

In this section we will suggest possible tools that can be used for testing myTaxiService application.

Of course the tools chosen depends on the programming language used; in this case we supposed that the application will be written in Java.

For the purpose of this integration test plan we will suggest to use *Arquillian* framework.
Thanks to its scenario-based structure and his IDE friendly attitude it will be simple to perform all the integration test listed.

5 Program Stubs and Test Data Required

In order to perform some integration tests we have identified three different stubs that simulate sample data needed:

- Client Stub: needed in test cases I0, I1, I2, I7
- Event Stub: needed in test case I0
- Booking Stub: needed in test case I3
- GPS Coordinates Stub: needed in test case I4

6 Appendix

6.1 Software and tool used

In order to redact this document we have used the following softwares:

- – Name: draw.io
  – Version: -
  – Scope: draw mudules diagram
  – Reference: `https://www.draw.io/`

6.2 Working Hours

The Integration Test Plan Document was written in more or less 16 hours divided between the two elements of the group:

- Massimiliano Paci: 8 hours
- Giovanni Patruno: 8 hours

Each element of the group didn't work on specific sections but worked on the whole document