Politecnico Di Milano
A.A. 2015/2016
Software Engineering 2:
”MyTaxiService”
Testing Document

Paramithiotti Andrea (Matr. 788794)
Rompani Andrea (Matr. 854052)
Zoia Lorenzo (Matr. 852392)

v1.0
21/01/2016

Contents

1 Introduction

1.1 Revision History

Version	date	Modifications
1.0	2016/1/21	First version

1.2 Purpose and Scope

1.2.1 Purpose

This document aims at describing how to plan the tests between all the components described in the Design Document in order to accomplish the integration. Every team member who cooperates in the integration test should read this document.

1.2.2 Scope

The aim of the project is to create a brand new system for the management and the organization of a city taxi service. This system offers a mobile application and a web interface in order to give the customers the possibility to benefit from the taxi service. Furthermore, the system provides an additional communication interface for the taxi drivers.
The mobile application and the web interface accept requests and reservations for taxis from the users, with the possibility to organise a share ride among different users. The taxi driver is supposed to communicate his availability, acceptances and rejections of requests through the communication interface.

The system is created to simplify the access of passengers to the service and to guarantee a fair management of the taxi queues.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Acronyms

- **DW:** Data Warehouse
- **DB:** Database
- **RASD:** Requirements Analysis and Specification Document
- **DD:** Design Document

1.4 Reference Documents

- The Project Description
- The RASD
- The DD

2 Integration Strategy

2.1 Entry Criteria

- The Mobile Application must be installable on the supported operating systems defined in the RASD
- The Mobile Application must have a working, stable and bidirectional connectivity with the system
- The Web Pages must be accessible from HTML 5 enabled web browsers
- The Taxi Driver Interface must have a working, stable and bidirectional connectivity with the system
- Security measures on the Users Database must be enforced as described in the DD and it must be accessible from within the dmz
- All the internal functionalities of each component must be unit tested

2.2 Elements to Be Integrated

In accordance with the DD, those elements are

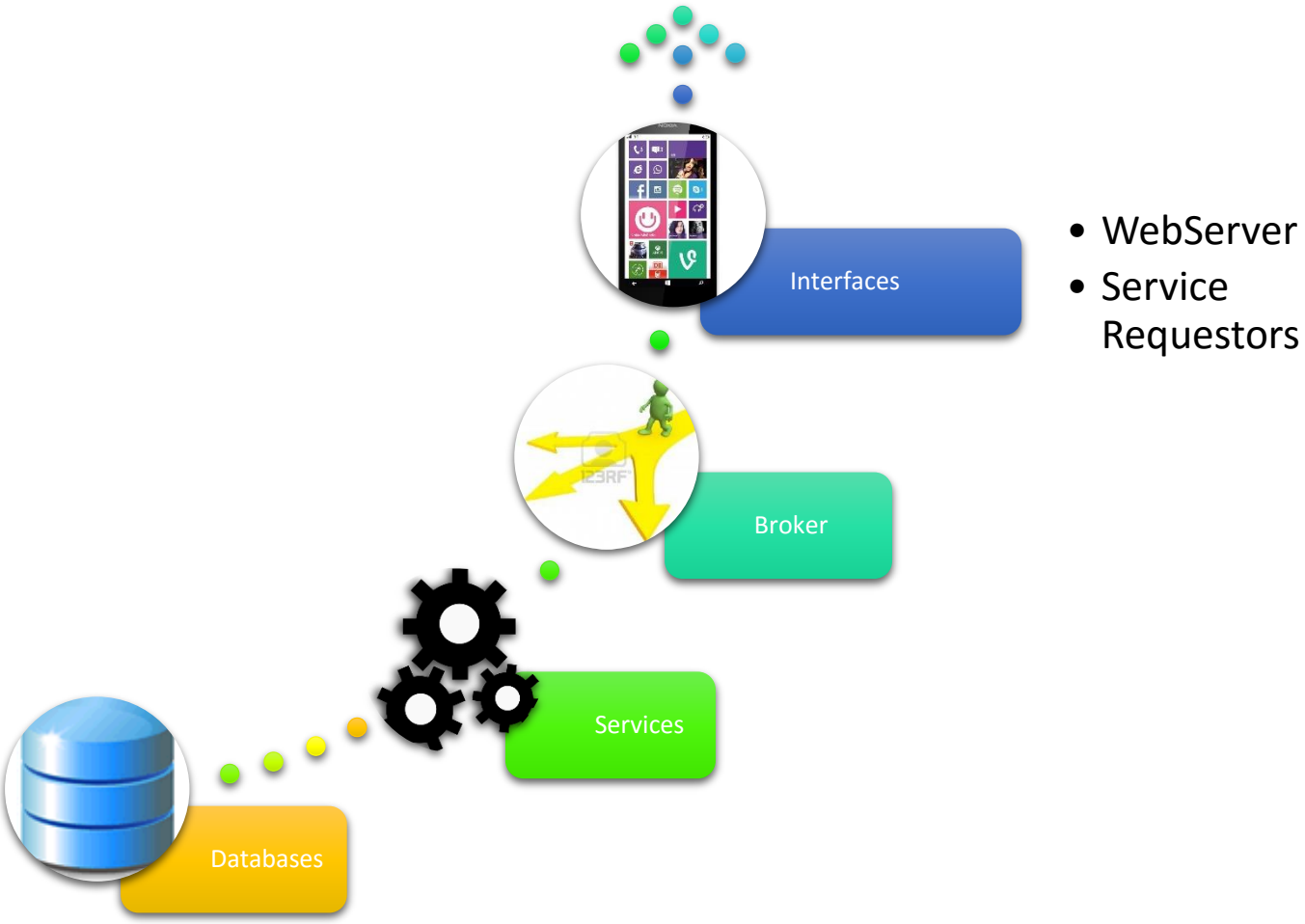
- Services
    - Authentication Service
    - Registration Service
    - Request Service
    - Reservation Service
    - Shared Ride Service
    - Emergency Service
    - Taxi Management Service
- Broker
    - Internal Message Dispatcher
    - Service Broker
    - Notification Center
- Web Server
- Service Requestors
    - Taxi Driver Interface
    - Mobile Interface
    - Web Interface
- Databases
    - Data Warehouse
    - Analyser
    - Users Database

2.3 Integration Testing Strategy

It was decided to follow a bottom-up approach according to the bottom-up design of the system.
The granularity of the design of the system forces a bottom-up strategy, otherwise a different approach would need to create a very large number of stubs and drivers, especially during the integration testing of the Broker components.

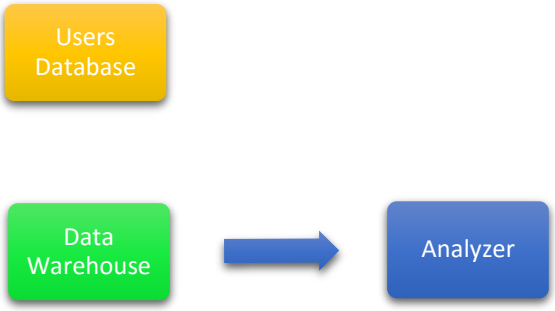
2.4 Sequence of Components Integration

2.4.1 Software Integration Systems



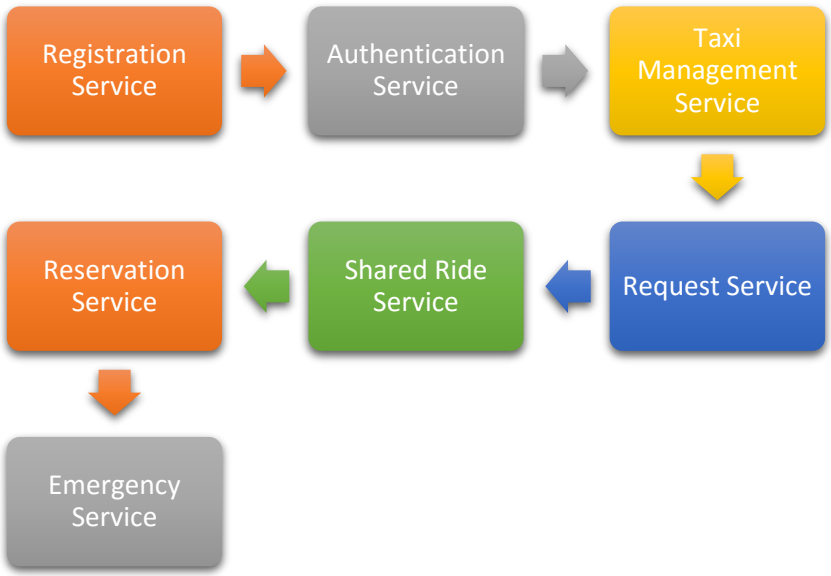
2.4.2 Subsystems Integration Sequence

2.4.2.1 Databases



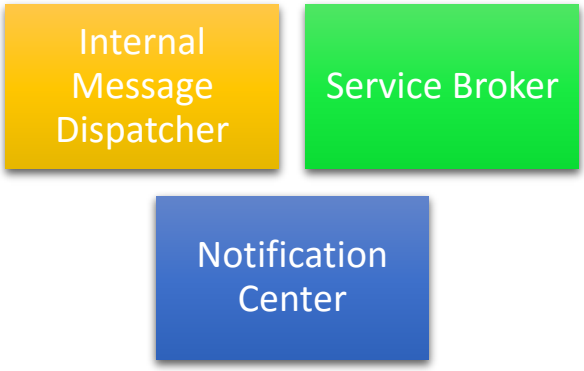
The Analyser is dependent on the Data Warehouse and must be tested only after the data warehouse.
There is no dependency between the Users Database and the data warehouse so their testing order is irrelevant.

2.4.2.2 Services



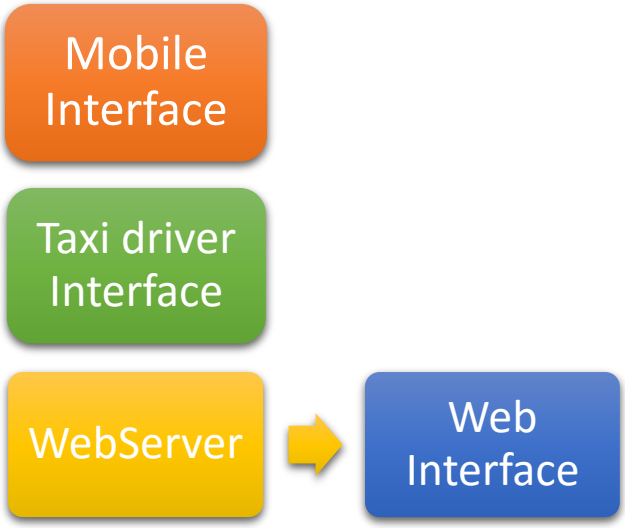
The Taxi management service is used by all the subsequent services so it must be tested before the others

2.4.2.3 Broker



No dependency between the components.

2.4.2.4 Interfaces



3 Individual Steps and Test Description

3.1 Integration Test Cases

3.1.1 Integration Test Case I1

Test Case Identifier	I1T1
Test Item	Authentication Service → User Database (Login)
Input Specification	Create typical Authentication Service input
Output Specification	Check if the correct methods are called in the User Database Interface and if all the data are stored correctly
Environmental Needs	Mock users data

Test Case Identifier	I1T2
Test Item	Registration service → User Database (RegisterUser)
Input Specification	Create typical Registration Service input
Output Specification	Check if the correct methods are called in the User Database Interface
Environmental Needs	Mock users data

3.1.2 Integration Test Case I2

Test Case Identifier	I2T1
Test Item	Service → Data Warehouse (Save Data)
Input Specification	Create typical save Request input
Output Specification	Check if the correct methods are called in the Data Warehouse Interface and all the data are stored correctly
Environmental Needs	N/A

Test Case Identifier	I2T2
Test Item	Analyser → Data Warehouse (Analyze)
Input Specification	Create typical Analyser input
Output Specification	Check if the correct methods are called in the Data Warehouse Interface
Environmental Needs	N/A

3.1.3 Integration Test Case I3

Test Case Identifier	I3T1
Test Item	User Interfaces → Registration Service (Registration)
Input Specification	Create typical registration input
Output Specification	Check if the correct methods are called in the Registration Service Interface
Environmental Needs	I1 succeeded

Test Case Identifier	I3T2
Test Item	User Interfaces → Authentication Service (Login)
Input Specification	Create typical authentication input
Output Specification	Check if the correct methods are called in the Authentication Service Interface
Environmental Needs	I1 succeeded

3.1.4 Integration Test Case I4

Test Case Identifier	I4T1
Test Item	Other Service→ Taxi Management (FindTaxi)
Input Specification	Create typical find taxi request input
Output Specification	Check if the correct methods are called in the Taxi Management Interface
Environmental Needs	I2 succeeded

Test Case Identifier	I4T2
Test Item	Taxi Interfaces → Taxi Management (SendTaxiAnswer)
Input Specification	Create typical taxi answer input
Output Specification	Check if the correct methods are called in the Taxi Management Interface
Environmental Needs	I2 succeeded

Test Case Identifier	I4T3
Test Item	Taxi Interfaces → Taxi Management (SetAvailability)
Input Specification	Create typical taxi availability input
Output Specification	Check if the correct methods are called in the Taxi Management Interface
Environmental Needs	I2 succeeded

3.1.5 Integration Test Case I5

Test Case Identifier	I5T1
Test Item	User Interfaces → Request Service (NewRequest)
Input Specification	Create typical request input
Output Specification	Check if the correct methods are called in the Request Service Interface
Environmental Needs	I4 succeeded

Test Case Identifier	I5T2
Test Item	Taxi Management → Request Service (LinkTaxi)
Input Specification	Create typical taxi information input
Output Specification	Check if the correct methods are called in the Request Service Interface
Environmental Needs	I4 succeeded

Test Case Identifier	I5T3
Test Item	Emergency Service → Request Service (AbortRequest)
Input Specification	Create typical request input
Output Specification	Check if the correct methods are called in the Request Service Interface
Environmental Needs	I4 succeeded

Test Case Identifier	I5T4
Test Item	Reservation Service → Request Service (StartReservation)
Input Specification	Create typical startReservation input
Output Specification	Check if the correct methods are called in the Request Service Interface
Environmental Needs	I4 succeeded

3.1.6 Integration Test Case I6

Test Case Identifier	I6T1
Test Item	User Interfaces → Reservation Service (ReserveTaxi)
Input Specification	Create typical reservation input
Output Specification	Check if the correct methods are called in the Reservation Service Interface
Environmental Needs	I2 succeeded

Test Case Identifier	I6T2
Test Item	User Interfaces →Reservation Service (CancelReservation)
Input Specification	Create typical reservation input
Output Specification	Check if the correct methods are called in the Reservation Service Interface
Environmental Needs	I2 succeeded

3.1.7 Integration Test Case I7

Test Case Identifier	I7T1
Test Item	User Interfaces → Shared Ride Service (StartSharedRide)
Input Specification	Create typical shared ride input
Output Specification	Check if the correct methods are called in the Shared Ride Service Interface
Environmental Needs	I5 succeeded

Test Case Identifier	I7T2
Test Item	User Interfaces → Shared Ride Service (FindCompatibleRide)
Input Specification	Create typical shared ride input
Output Specification	Check if the correct methods are called in the Shared Ride Service Interface
Environmental Needs	I5 succeeded

3.1.8 Integration Test Case I8

Test Case Identifier	I8T1
Test Item	User Interfaces → Emergency Service (EndRide)
Input Specification	Create typical ride information input
Output Specification	Check if the correct methods are called in the Emergency Service Interface
Environmental Needs	I5 succeeded

Test Case Identifier	I8T2
Test Item	User Interfaces → Emergency Service (SendNewTaxi)
Input Specification	Create typical new taxi request input
Output Specification	Check if the correct methods are called in the Emergency Service Interface
Environmental Needs	I4 succeeded

3.1.9 Integration Test Case I9

Test Case Identifier	I9T1
Test Item	User Interfaces → Service Broker (Dispatch message)
Input Specification	Create typical message input
Output Specification	Check if the correct methods are called in the Service Broker Interface
Environmental Needs	N/A

3.1.10 Integration Test Case I10

Test Case Identifier	I10T1
Test Item	Services → Internal Message Dispatcher (Dispatch message)
Input Specification	Create typical internal message input
Output Specification	Check if the correct methods are called in the Internal Message Dispatcher Interface
Environmental Needs	I6, I7, I8, I9 succeded

3.1.11 Integration Test Case I11

Test Case Identifier	I11T1
Test Item	Services → Notification Center (SendNotification)
Input Specification	Create typical notification request input
Output Specification	Check if the correct methods are called in the Notification Center Interface
Environmental Needs	Notification subsystem stub and driver, web server driver

3.1.12 Integration Test Case I12

Test Case Identifier	I12T1
Test Item	Web Interface → Web Server
Input Specification	Create typical html request input
Output Specification	Check if the correct methods are called in the Web Server and if are displayed the right pages
Environmental Needs	I11 succeeded

3.2 Integration Test Procedure

3.2.1 Integration Test Procedure TP1

Test Procedure Identifier	TP1
Purpose	This test procedure verifies whether the User Database • Can handle a registration request • Can handle an authentication request
Procedure Steps	Execute all the subcases of I1

3.2.2 Integration Test Procedure TP2

Test Procedure Identifier	TP2
Purpose	This test procedure verifies whether the Data Warehouse • Can handle a store data request • Can handle an access data request
Procedure Steps	Execute all the subcases of I2

3.2.3 Integration Test Procedure TP3

Test Procedure Identifier	TP3
Purpose	This test procedure verifies whether the Authentication Service • Can handle an authentication request from an user In addition verifies whether the Registration Service • Can handle a registration request from an user
Procedure Steps	Execute all the subcases of I3

3.2.4 Integration Test Procedure TP4

Test Procedure Identifier	TP4
Purpose	This test procedure verifies whether the Taxi Management Service • Can handle a searching taxi request from another service • Can handle an availability answer sent by a taxi driver • Can handle an availability changing request sent by a taxi driver
Procedure Steps	Execute all the subcases of I4

3.2.5 Integration Test Procedure TP5

Test Procedure Identifier	TP5
Purpose	This test procedure verifies if the system is able to handle any requests from the user. In details, it verifies whether the Request Service • Can handle a create new request input • Can handle a link taxi request received from the Taxi Management Service • Can handle an abort ride input • Can handle a start reservation request Whether the Reservation Service • Can handle a create reservation procedure • Can handle a cancel reservation request Whether the Shared Ride Service • Can handle the start shared ride command • Can handle the find compatible ride request
Procedure Steps	Execute I5 before I6-I7

3.2.6 Integration Test Procedure TP6

Test Procedure Identifier	TP6
Purpose	This test procedure verifies whether the Emergency Service • Can handle an end ride request • Can handle send new taxi request
Procedure Steps	Execute all the subcases of I8

3.2.7 Integration Test Procedure TP7

Test Procedure Identifier	TP7
Purpose	This test procedure verifies if all the components of the Broker can execute the User-Services and Services-Services in the right way. In details, it verifies whether the Service Broker • Can dispatch to the right service any possible message from the user And whether the Internal Message Dispatcher • Can dispatch to the right service any possible message from any service And whether the Notification Center • Can send the right notification to the user
Procedure Steps	Execute I9-I11

3.2.8 Integration Test Procedure TP8

Test Procedure Identifier	TP8
Purpose	This test procedure verifies whether the Web Server • Can handle any html request sent from the web interface
Procedure Steps	Execute all the subcases of I12

4 Tools and Test Equipment Required

Here is the list of the tools and test equipment useful for the purpose. We assume that the application is written in JEE, as most of the tools available on the web can be used with this language.

4.1 H2 Database Engine

http://www.h2database.com/html/main.html

This open source tool provides a browser based interface that allows the user to create a stub database in order to test the interaction with other components. In this way it is not necessary to use the real database.

4.2 Mockito

http://mockito.github.io

Tool for the creation of mockups for unit testing. It provides a framework for interaction testing. It can be used for the creation of the stubs.

4.3 JMeter

http://jmeter.apache.org

A GUI desktop application designed to load test functional behavior and measure performance. It can be used also to simulate the traffic on the server and to stress test it.

4.4 Manual Testing

Manual testing can be useful in the cases where the use of a tool is more expensive in terms of time or effort. In this case the TP1 and TP2 can be performed with manual testing, as they consist in SQL testing on the database of the application.

5 Program Stubs and Test Data Required

- For test case I1 the database must be already filled with some mock users data
- No particular stub is needed for the integration testing due to the bottom up design
- Drivers are needed for the creation of the inputs of the test cases so they are omitted from the test cases tables to avoid repetition
- A notification center driver for test cases I3 to I8 is needed to receive the notification messages