Integration Test Plan Document

MyTaxiService

Pozzi Matteo

Scandroglio Stefano

1 Introduction

Integration testing is the phase in the software testing process in which individual software modules are combined and tested as a group.
The purpose of this phase is to verify that functional and non-functional requirements, specified in previous phases of the software development process, are met.

1.1 Revision History

This represents the first version of this document.

1.2 Purpose and Scope

The purpose of this document is to describe the plans for testing the integration of the different components of the MyTaxiService application that we have identified in the Design Document.
MyTaxiService is an application that allows users to request and reserve taxi rides, drivers to either accept or refuse those requests. The application will also manage different queues of taxis, each one associated to a particular zone inside the city to which this application is destined.

1.3 Definitions and Abbreviations

Here is an explanation of all the specific terms and abbreviations used in this document.

1.3.1 Definitions

- **User** : a registered person who is able to requests and reserve a taxi;
- **Driver** : the person driving the taxi;
- **Request** : when a user asks to be picked up by a taxi as soon as possible;
- **Zone** : a 2 km^2 part of the city;
- **Stub** : a piece of software that simulates the behavior of another piece which is not yet tested;
- **Driver** : a piece of software that simulates a call to the piece of software under test.

1.3.2 Abbreviations

- **RASD** : Requirement Analysis and Specification Document;
- **DD** : Design Document;
- **TC** : Test Case.

1.4 List of reference documents

To properly understand this document we recommend to previously read the Requirements and Specification Document (RASD) and the Design Document (DD) that we already presented.

2 Integration Strategy

2.1 Entry criteria

Before integration testing can begin we suppose that the following conditions hold:

- RASD has been delivered
- DD has been delivered
- Single software modules have been fully developed and

2.2 Elements to be integrated

We have identified the elements to be integrated from the Components Diagram in the DD, which are:

- Authentication
- Driver Manager
- User Manager
- Account Settings
- Zone
- Request
- Notification

The functions of these components are described in paragraph 2.2 of DD.

2.3 Integration testing strategy

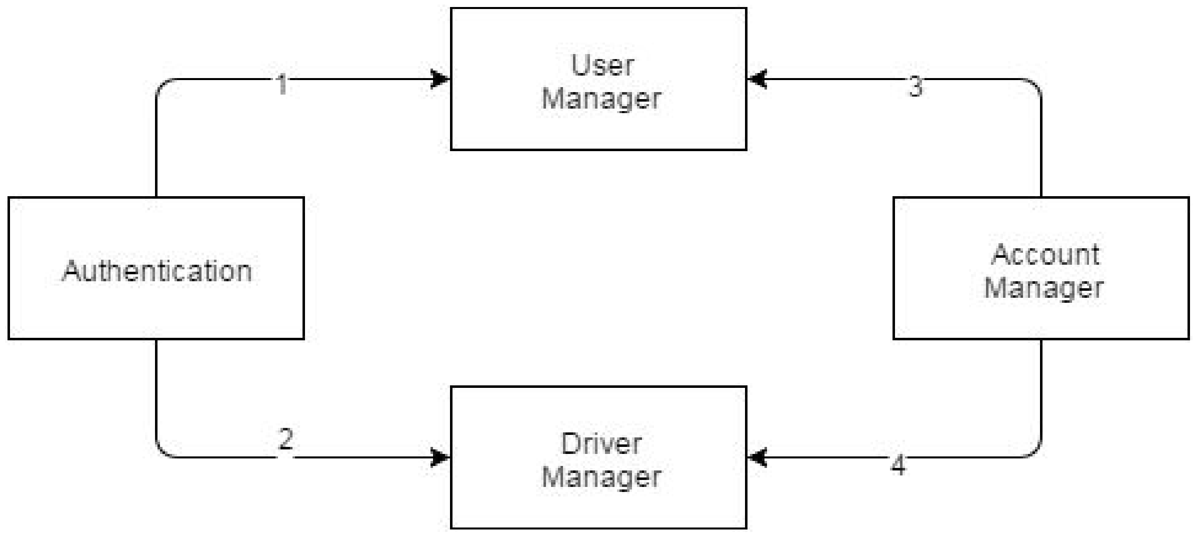
The integration strategy that we have decided to use is the functional groupings: components which collaborate to provide the same functionality are integrated and tested. We have decided to choose this approach so that we can first ensure that the single functionalities work as they are supposed to and then, we can ensure that the system as a whole works as intended.

2.4 Sequence of component integration

This section describes the sequence of component integration and the precedences according to which components/subsystem will be integrated (represented by arrows).

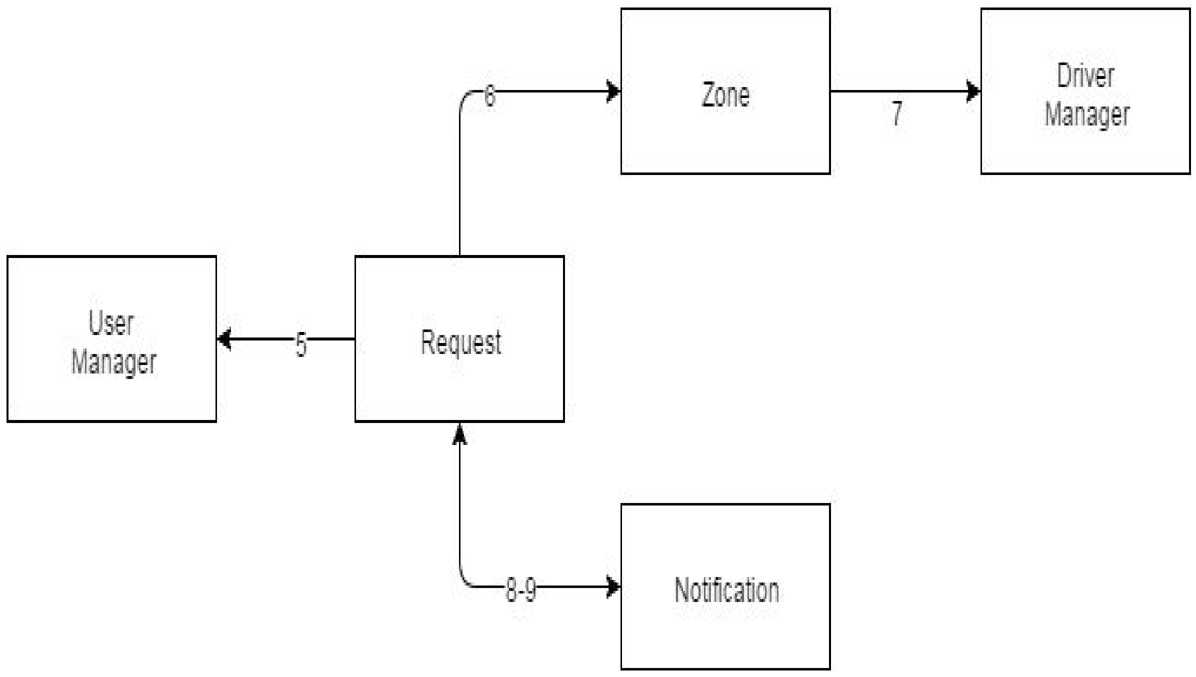
2.4.1 Software integration sequence

The two diagrams below shows the integration and testing order inside the **Authentication Subsystem** and the different test cases described in the next chapters:



ID	Integration Test	Paragraphs
TC1	Authentication → User Manager	3.1 5.2
TC2	Authentication → Driver Manager	3.2 5.2
TC3	Account Manager → User Manager	3.3 5.2
TC4	Account Manager → Driver Manager	3.4 5.2

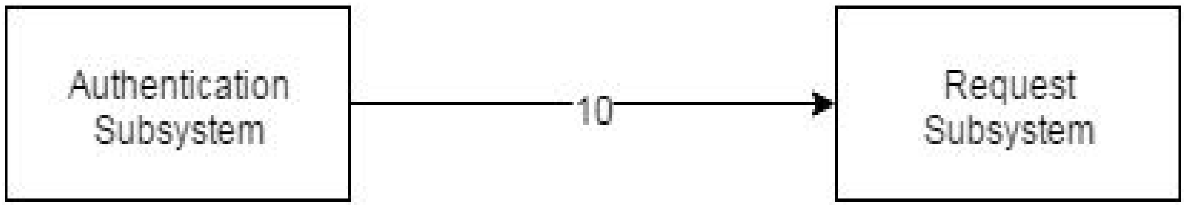
The two diagrams below shows the integration and testing order inside the **Request Subsystem** and the different test cases described in the next chapters:



ID	Integration Test	Paragraphs
TC5	Request → User Manager	3.5 5.2
TC6	Request → Zone	3.6 5.1
TC7	Zone → Driver Manager	3.7
TC8	Request → Notification	3.8
TC9	Notification → Request	3.9 5.1 5.2

2.4.2 Subsystem integration sequence

The diagram below shows the integration and testing order among the two subsystems described in the previous chapter:



ID	Integration Test	Paragraphs
TC10	Authentication Subsystem → Request Subsystem	3.10

3 Individual Steps and Test Description

3.1 Integration Test Case 1

Test Case Identifier	TC1
Test Item(s)	Authentication → User Manager
Input Specification	User login/registration data
Output Specification	Check if correct functions are called in User Manager
Environmental Needs	User driver

3.2 Integration Test Case 2

Test Case Identifier	TC2
Test Item(s)	Authentication → Driver Manager
Input Specification	Driver login/registration data
Output Specification	Check if correct functions are called in Driver Manager
Environmental Needs	Driver driver

3.3 Integration Test Case 3

Test Case Identifier	TC3
Test Item(s)	Account Manager → User Manager
Input Specification	New user account’s data
Output Specification	Check if correct functions are called in User Manager
Environmental Needs	TC1 succeeded, User driver

3.4 Integration Test Case 4

Test Case Identifier	TC4
Test Item(s)	Account Manager → Driver Manager
Input Specification	New driver account’s data
Output Specification	Check if correct functions are called in Driver Manager
Environmental Needs	TC2 succeeded, Driver driver

3.5 Integration Test Case 5

Test Case Identifier	TC5
Test Item(s)	Request → User Manager
Input Specification	Request made by the user
Output Specification	Check if correct functions are called in User Manager
Environmental Needs	User driver

3.6 Integration Test Case 6

Test Case Identifier	TC6
Test Item(s)	Request → Zone
Input Specification	New request made by the user
Output Specification	Check if correct functions are called in Zone component
Environmental Needs	User driver

3.7 Integration Test Case 7

Test Case Identifier	TC7
Test Item(s)	Zone → Driver Manager
Input Specification	Zone from which the request has been made
Output Specification	Check if correct functions are called in Driver Manager
Environmental Needs	TC6 succeeded

3.8 Integration Test Case 8

Test Case Identifier	TC8
Test Item(s)	Request → Notification
Input Specification	Driver data to send him/her the request
Output Specification	Check if correct functions are called in Notification component
Environmental Needs	TC7 succeeded

3.9 Integration Test Case 9

Test Case Identifier	TC9
Test Item(s)	Notification → Request
Input Specification	Accepted/Refused request by the driver
Output Specification	Check if correct functions are called by Request component
Environmental Needs	TC8 succeeded, Driver driver

3.10 Integration Test Case 10

Test Case Identifier	TC10
Test Item(s)	Authentication Subsystem → Request Subsystem
Input Specification	User and Driver authentication data
Output Specification	Check if correct functions are called in Request Subsystem
Environmental Needs	User driver, Driver driver

4 Tools and Test Equipment Required

To perform the test cases specified in this document we recommend using the following tools:

- **JUnit** (junit.org) : JUnit is a unit testing framework for the Java programming language;

- **Mockito** (mockito.org) : open source testing framework for Java released under the MIT License;

- **Arquillian** (arquillian.org) : Arquillian is an innovative and highly extensible testing platform for the JVM that enables developers to easily create automated integration, functional and acceptance tests for Java middleware.

5 Program Stubs and Test Data Required

In this section we describe the stubs and drivers required to perform the test cases we have previously identified.

5.1 Stubs

TC6 :

- a method that retrieves the head of a certain zone (e.g. getHead())
- it will be tested in **TC7**

TC9 :

- a method that updates the status of the driver who has accepted the request
- a method that notifies the user who has made the request that his request has been accepted

5.2 Drivers

TC1 :

- user who inputs login/registration data

TC2 :

- driver who inputs login/registration data

TC3 :

- user who inputs account modification data

TC4 :

- driver who inputs account modification data

TC5 :

- user who wants to make a new request

TC9 :

- driver who either accepts or refuses the request through the notification