Politecnico di Milano
A.A. 2015-2016
Software Engineering 2

Test Plan

Giuseppe Canonaco, Andrea Di Giosaffatte

21 January 2016 – Version 1.1

Table of Contents

1. Introduction

1.1. Revision History 3
1.2. Purpose and Scope 3
1.3. List of Definitions and Abbreviations 3
1.4. List of Reference Documents 3

2. Integration Strategy

2.1. Entry Criteria 3
2.2. Elements to be Integrated 4
2.3. Integration Testing Strategy 5
2.4. Sequence of Component Integration 5
2.4.1. Integration Sequence 5

3. Individual Steps and Test Description

3.1. Individual Steps and Test Description 6

4. Tools and Test Equipment Required

4.1. Tools and Test Equipment Required 18

5. Program Stubs and Drivers

5.1. Program Stubs and Drivers 19

6. Used Tools

6.1. Used Tools 20

1. Introduction

1.1. Revision History

- Integration test plan v1.0 (current)

1.2. Purpose and Scope

This document describes the plan for testing the integration of all the components designed into the DD. In other words, it aims at describing how to test the interfaces among all the components developed in the MyTaxiService project to see if the interaction by means of them is properly working.

This document is a compulsory guideline to carry out the integration test phase.

1.3. List of Definitions and Abbreviations

- **DD:** MyTaxiService Design Document
- **RASD:** MyTaxiService RASD Document

1.4. Reference Documents

- MyTaxiService RASD Document
- MyTaxiService Design Document
- Assignment 4 (Test Plan) Document
- Documentation of the testing tool proposed to cope with the integration test phase

  [1] https://docs.jboss.org/arquillian/reference/1.0.0.Alpha1/en-US/html_single/

2. Integration Strategy

2.1. Entry Criteria

- All the internal documents (RASD and DD) are up to date and they reflect the current state of the project.

- All the needed testing tools must be already prepared.

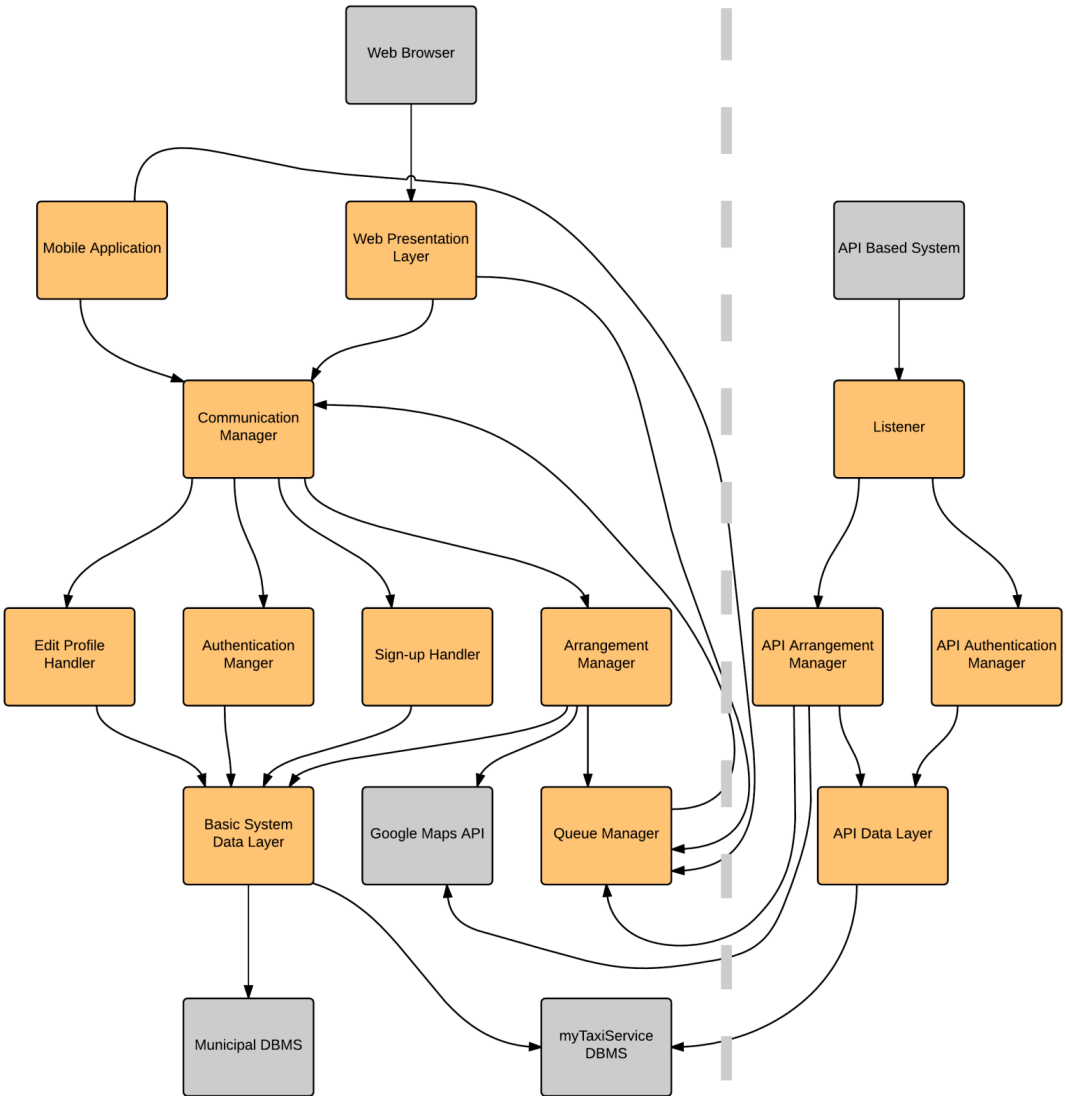
- Before starting each integration test, the components involved in that test must be code complete and unit tested.

2.2. Elements to be Integrated

In the following diagram are represented in orange all the components which need to be integrated and in light grey the components which use or are used by the components to be tested. This last set of components are the Web Browser and the API Based System, which are mocked by tailored drivers in the last stage of the integration, the Municipal DBMS, the MyTaxiService DBMS and the Google maps API, which are just used by the components to be tested.

The arrows represent the use relationship among the components connected by them: the head highlights the used component and vice-versa with respect to the tail.

Diagram 1



2.3. Integration Testing Strategy

To approach the integration test phase it is mandatory following a bottom-up strategy, the only exception is the queue manager, which needs a stub due to its dependencies.
This strategy was chosen because doesn't require too much effort in planning, which is the reason why the functional together with the sandwich approach were discarded. The top-down approach was, at first, taken into account, but then discarded because is mainly used when the project needs to speed up this phase due to deadline constraints, indeed, with this strategy the low level components are roughly tested.

2.4. Sequence of Component Integration

There are two subsystems which need to be integrated: the API subsystem and the basic application subsystem. The former is composed of the Listener, the API Arrangement Manager, the API Authentication Manager and the API Data Layer; the latter is composed of the remaining orange components depicted in the diagram 1 above in the document. These two subsystems interact through the interface offered by the Queue manager and their integration is not performed after they are, separately, internally integrated, indeed, this would need a superfluous stub, the one mocking the API Arrangement Manager.

2.4.1. Integration Sequence

I1-T1	Basic System Data Layer → Municipal DBMS
I1-T2	Basic System Data Layer → MyTaxiService DBMS
I5-T1	Queue Manager → Communication Manager
I10-T1	API Data Layer → MyTaxiService DBMS
I2-T1	Edit Profile Handler → Basic System Data Layer
I3-T1	Authentication Manager → Basic System Data Layer
I4-T1	Sign-up Handler → Basic System Data Layer
I6-T1	Arrangement Manager → Basic System Data Layer
I6-T2	Arrangement Manager → Queue Manager
I6-T3	Arrangement Manager → Google Maps API
I11-T1	API Authentication Manager → API Data Layer
I12-T2	API Arrangement Manager → Queue Manager
I12-T3	API Arrangement Manager → Google Maps API
I7-T1	Communication Manager → Edit Profile Handler
I7-T2	Communication Manager → Authentication Manager
I7-T3	Communication Manager → Sign-up Handler
I7-T4	Communication Manager → Arrangement Manager
I13-T1	Listener → API Arrangement Manager
I13-T2	Listener → API Authentication Manager
I8-T1	Web Presentation Layer → Communication Manager
I8-T2	Web Presentation Layer → Queue Manager
I9-T1	Mobile Application → Communication Manager
I9-T2	Mobile Application → Queue Manager

Each integration test block must be performed one after the other, but, internally, the block's integration tests can be executed in parallel to speed up this phase. For a more comprehensive view of each integration test have a look at section Individual Steps and Test Description of this document.

3. Individual Steps and Test Description

3.1. Individual Steps and Test Description

In this section is described how to design all the tests that will be used to verify that the elements integrated perform as expected. Each test will be done in order to avoid the following integration faults:

- Inconsistent interpretation of parameters or values.
- Violations of value domains, capacity, or size limits.
- Side effects on parameters or resources.
- Omitted or misunderstood functionality.
- Non-functional properties.
- Dynamic mismatches.

I1: Basic System Data Layer

Integration Test Identifier: I1-T1.1

Components Involved in the Test: Basic System Data Layer → Municipal DBMS

Environmental Needs: Driver representing the Edit Profile Handler component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab driver to edit his profile.

Expected Result: The invoked methods call properly the right methods offered by the Municipal DBMS interface in order to guarantee the profile editing functionality.

Integration Test Identifier: I1-T1.2

Components Involved in the Test: Basic System Data Layer → Municipal DBMS

Environmental Needs: Driver representing the Authentication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab driver to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the Municipal DBMS interface in order to guarantee the authentication functionality.

Integration Test Identifier: I1-T2.1

Components Involved in the Test: Basic System Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the Arrangement Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the arrangement management functionalities, which are, according to the DD, store reservation (save a reservation over the data base) and delete reservation (delete a reservation not yet in handling).

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee all the arrangement management functionalities.

Integration Test Identifier: I1-T2.2

Components Involved in the Test: Basic System Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the Edit Profile Handler component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab catcher to edit his profile.

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee the profile editing functionality.

Integration Test Identifier: I1-T2.3

Components Involved in the Test: Basic System Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the Authentication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab catcher to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee the authentication functionality.

Integration Test Identifier: I1-T2.4

Components Involved in the Test: Basic System Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the Sign Up Handler component.

Test Criterion: Through the driver, we invoke methods offered by the Basic System Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a new cab catcher to register over the system.

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee the sign up functionality.

I2: Edit Profile Handler

Integration Test Identifier: I2-T1.1

Components Involved in the Test: Edit Profile Handler Basic → System Data Layer

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Profile Editing interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to edit his profile.

Expected Result: The invoked methods call properly the right methods offered by the Basic System Data Layer interface in order to guarantee the profile editing functionality.

I3: Authentication Handler

Integration Test Identifier: I3-T1.1

Components Involved in the Test: Authentication Handler → Basic System Data Layer

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Authentication interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the Basic System Data Layer interface in order to guarantee the authentication functionality.

I4: Sign Up Handler

Integration Test Identifier: I4-T1.1

Components Involved in the Test: Sign Up Handler → Basic System Data Layer

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Sign Up interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a new cab catcher to register over the system.

Expected Result: The invoked methods call properly the right methods offered by the Basic System Data Layer interface in order to guarantee the sign up functionality.

I5: Queue Manager

Integration Test Identifier: I5-T1.1

Components Involved in the Test: Queue Manager → Communication Manager

Environmental Needs: Driver representing the Arrangement Manager component; Stub representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Cab Driver Allocation Request interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the cab driver allocation functionalities, which are, according to the DD, allocate cab driver (dispatch a cab driver which will handle an arrangement) and delete dispatch (stop the dispatch procedure or, if the cab driver was already dispatched, switch him from the unavailable-cab-driver-queue to the available one).

Expected Result: The invoked methods call properly the right methods on the Communication Manager stub.

Integration Test Identifier: I5-T1.2

Components Involved in the Test: Queue Manager → Communication Manager

Environmental Needs: Driver representing the API Arrangement Manager component; Stub representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Cab Driver Allocation Request interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the cab driver allocation functionalities, which are, according to the DD, allocate cab driver (dispatch a cab driver which will handle an arrangement) and delete dispatch (stop the dispatch procedure or, if the cab driver was already dispatched, switch him from the unavailable-cab-driver-queue to the available one).

Expected Result: The invoked methods call properly the right methods on the Communication Manager stub.

I6: Arrangement Manager

Integration Test Identifier: I6-T1.1

Components Involved in the Test: Arrangement Manager → Basic System Data Layer

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Handle Arrangement interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the arrangement management functionalities, which are, according to the DD, handle new arrangement and delete arrangement.

Expected Result: The invoked methods call properly the right methods offered by the Basic System Data Layer interface in order to guarantee all the arrangement management functionalities.

Integration Test Identifier: I6-T2.1

Components Involved in the Test: Arrangement Manager → Queue Manager

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Handle Arrangement interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the arrangement management

functionalities, which are, according to the DD, handle new arrangement and delete arrangement.

Expected Result: The invoked methods call properly the right methods offered by the Cab Driver Allocation Request interface in order to guarantee all the arrangement management functionalities.

Integration Test Identifier: I6-T3.1

Components Involved in the Test: Arrangement Manager → Google Maps API

Environmental Needs: Driver representing the Communication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the Handle Arrangement interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the handle arrangement functionality.

Expected Result: The invoked methods call properly the right methods offered by the Google Maps API interface in order to guarantee the handle arrangement functionality.

I7: Communication Manager

Integration Test Identifier: I7-T1.1

Components Involved in the Test: Communication Manager → Edit Profile Handler

Environmental Needs: Driver representing the Mobile Application component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to edit his profile.

Expected Result: The invoked methods call properly the right methods offered by the Profile Editing interface in order to guarantee the profile editing functionality.

Integration Test Identifier: I7-T1.2

Components Involved in the Test: Communication Manager → Edit Profile Handler

Environmental Needs: Driver representing the Web Presentation Layer component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to edit his profile.

Expected Result: The invoked methods call properly the right methods offered by the Profile Editing interface in order to guarantee the profile editing functionality.

Integration Test Identifier: I7-T2.1

Components Involved in the Test: Communication Manager → Authentication Manager

Environmental Needs: Driver representing the Mobile Application component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the Authentication interface in order to guarantee the authentication functionality.

Integration Test Identifier: I7-T2.2

Components Involved in the Test: Communication Manager → Authentication Manager

Environmental Needs: Driver representing the Web Presentation Layer component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a user to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the Authentication interface in order to guarantee the authentication functionality.

Integration Test Identifier: I7-T3.1

Components Involved in the Test: Communication Manager → Sign Up Handler

Environmental Needs: Driver representing the Mobile Application component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the sign up functionality that allows a new cab catcher to register over the system.

Expected Result: The invoked methods call properly the right methods offered by the Sign Up interface in order to guarantee the sign up functionality.

Integration Test Identifier: I7-T3.2

Components Involved in the Test: Communication Manager → Sign Up Handler

Environmental Needs: Driver representing the Web Presentation Layer component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the sign up functionality that allows a new cab catcher to register over the system.

Expected Result: The invoked methods call properly the right methods offered by the Sign Up interface in order to guarantee the sign up functionality.

Integration Test Identifier: I7-T4.1

Components Involved in the Test: Communication Manager → Arrangement Manager

Environmental Needs: Driver representing the Mobile Application component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the arrangement management functionalities, which are, according to the DD, taxi request (make a taxi request), reservation (make a taxi reservation) and delete arrangement (delete an arrangement previously asked).

Expected Result: The invoked methods call properly the right methods offered by the Handle Arrangement interface in order to guarantee all the arrangement management functionalities.

Integration Test Identifier: I7-T4.2

Components Involved in the Test: Communication Manager → Arrangement Manager

Environmental Needs: Driver representing the Web Presentation Layer component.

Test Criterion: Through the driver, we invoke methods offered by the Basic Application interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the arrangement management functionalities, which are, according to the DD, taxi request (make a taxi request), reservation (make a taxi reservation) and delete arrangement (delete an arrangement previously asked).

Expected Result: The invoked methods call properly the right methods offered by the Handle Arrangement interface in order to guarantee all the arrangement management functionalities.

I8: Web Presentation Layer

Integration Test Identifier: I8-T1.1

Components Involved in the Test: Web Presentation Layer → Communication Manager

Environmental Needs: Driver representing the Web Browser component.

Test Criterion: Through the driver, we invoke methods offered by the Web Browser interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that allow a user to surf the web pages (representing the web application) and of those that collect the information related to the actions performed over the web pages.

Expected Result: The invoked methods call properly the right methods offered by the Basic Application interface in order to guarantee all the functionalities related with a web user.

Integration Test Identifier: I8-T2.1

Components Involved in the Test: Web Presentation Layer → Queue Manager

Environmental Needs: Driver representing the Web Browser component.

Test Criterion: Through the driver, we invoke methods offered by the Web Browser interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that allow a cab driver to send information about availability status updates and answers to arrangement assignations.

Expected Result: The invoked methods call properly the right methods offered by the Update Cab Driver Position interface in order to guarantee all the functionalities related with the status of a cab driver.

I9: Mobile Application

Integration Test Identifier: I9-T1.1

Components Involved in the Test: Mobile Application → Communication Manager

Environmental Needs: Driver emulating the end user.

Test Criterion: Through the driver, we invoke methods offered by the Mobile Application interface (GUI), mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that collect the information related to the actions performed by the user over the mobile application.

Expected Result: The invoked methods call properly the right methods offered by the Basic Application interface in order to guarantee all the functionalities related with a mobile user.

Integration Test Identifier: I9-T2.1

Components Involved in the Test: Mobile Application → Queue Manager

Environmental Needs: Driver emulating the end user.

Test Criterion: Through the driver, we invoke methods offered by the Mobile Application interface (GUI), mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that allow a cab driver to send information about availability status updates and answers to arrangement assignations.

Expected Result: The invoked methods call properly the right methods offered by the Update Cab Driver Position interface in order to guarantee all the functionalities related with the status of a cab driver.

I10: API Data Layer

Integration Test Identifier: I10-T1.1

Components Involved in the Test: API Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the API Authentication Manager component.

Test Criterion: Through the driver, we invoke methods offered by the API Manager Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab catcher from an API based system to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee the API authentication functionality.

Integration Test Identifier: I10-T1.2

Components Involved in the Test: API Data Layer → MyTaxiService DBMS

Environmental Needs: Driver representing the API Arrangement Manager component.

Test Criterion: Through the driver, we invoke methods offered by the API Manager Data Layer interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the API arrangement management functionalities, which are, according to the DD, store reservation (save a reservation from an API based system over the data base) and delete reservation (delete a reservation not yet in handling).

Expected Result: The invoked methods call properly the right methods offered by the MyTaxiService DBMS interface in order to guarantee all the API arrangement management functionalities.

I11: API Authentication Manager

Integration Test Identifier: I11-T1.1

Components Involved in the Test: API Authentication Manager → API Data Layer

Environmental Needs: Driver representing the Listener component.

Test Criterion: Through the driver, we invoke methods offered by the API Authentication Manager interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the functionality that allows a cab catcher from an API based system to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the API Manager Data Layer interface in order to guarantee the API authentication functionality.

I12: API Arrangement Manager

Integration Test Identifier: I12-T1.1

Components Involved in the Test: API Arrangement Manager → API Data Layer

Environmental Needs: Driver representing the Listener component.

Test Criterion: Through the driver, we invoke methods offered by the API Arrangement Manager interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the API arrangement management functionalities, which are, according to the DD, handle new arrangement and delete arrangement.

Expected Result: The invoked methods call properly the right methods offered by the API Manager Data Layer interface in order to guarantee the API arrangement management functionalities.

Integration Test Identifier: I12-T2.1

Components Involved in the Test: API Arrangement Manager → Queue Manager

Environmental Needs: Driver representing the Listener component.

Test Criterion: Through the driver, we invoke methods offered by the API Arrangement Manager interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the API arrangement management functionalities, which are, according to the DD, handle new arrangement and delete arrangement.

Expected Result: The invoked methods call properly the right methods offered by the Cab Driver Allocation Request interface in order to guarantee the API arrangement management functionalities.

Integration Test Identifier: I12-T3.1

Components Involved in the Test: API Arrangement Manager → Google Maps API

Environmental Needs: Driver representing the Listener component.

Test Criterion: Through the driver, we invoke methods offered by the API Arrangement Manager interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the handle arrangement functionality.

Expected Result: The invoked methods call properly the right methods offered by the Google Maps API interface in order to guarantee the handle arrangement functionality.

I13: Listener

Integration Test Identifier: I13-T1.1

Components Involved in the Test: Listener → API Arrangement Manager

Environmental Needs: Driver representing the API Based System component.

Test Criterion: Through the driver, we invoke methods offered by the Programmatic interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked methods is composed of those that are related to the API arrangement management functionalities, which are, according to the DD, request (allows an API based system to yield a request), reservation (allows the API based system to make a reservation), delete arrangement (allows an API based system to delete an arrangement) and check status (allows an API based system to check the status of a particular arrangement).

Expected Result: The invoked methods call properly the right methods offered by the API Arrangement Manager interface in order to guarantee all the API arrangement management functionalities.

Integration Test Identifier: I13-T2.1

Components Involved in the Test: Listener → API Authentication Manager

Environmental Needs: Driver representing the API Based System component.

Test Criterion: Through the driver, we invoke methods offered by the Programmatic interface, mostly taking into account all the (non-trivial) edge cases. The set of the invoked

methods is composed of those that are related to the functionality that allows a cab catcher from an API based system to authenticate.

Expected Result: The invoked methods call properly the right methods offered by the API Authentication Manager interface in order to guarantee the API authentication functionality.

4. Tools and Test Equipment Required

4.1. Tools and Test Equipment Required

Since JEE is the language used for the developing phase, to cope with the integration testing phase is needed a particular testing framework, which is Arquillian. It strives to make integration testing no more complicated than basic unit testing and this is the main reason why it has been chosen. This testing framework will be used for instance to “ensure that the declarative services, such as dependency injection, actually get applied and work as expected. What happens when your Message Driven Bean can't parse the XML message? Will the right component be injected?”[1].

Arquillian must be configured with java and jUnit and this is why jUnit is required as a tool to accomplish the procedures described by this document (the minimum requirement on jUnit imposed by Arquillian is version 4.8).

The desktop testing environment is composed of the subsequent tools:

- Glassfish server open source edition 4.1.1
- Oracle database 12c
- Java EE 7
- NetBeans IDE 8.1
- Ubuntu 15.1
- Latest version of Google Chrome, Mozilla Firefox, Safari, Opera and Internet Explorer to test the web application.

The mobile application must be tested over the last version of iOS and on Android Gingerbread v2.3.3-2.3.7, Ice Cream Sandwich v4.0.3-4.0.4 and Jelly Bean v4.1.x-4.2.x.

5. Program Stubs and Drivers

5.1. Program Stubs and Drivers

In order to properly implement the bottom-up testing strategy some test drivers are needed. Test Drivers are used to simulate the behavior of the upper level components that are not yet integrated, indeed, they act as temporary replacement for a calling component and give the same output as that of the actual product.

The components that need to be simulated by a test driver are:

- Edit Profile Handler
- Authentication Manager
- Sign Up Handler
- Arrangement Manager
- Communication Manager
- Mobile Application
- Web Presentation Layer
- Web Browser
- API Authentication Manager
- API Arrangement Manager
- Listener
- API Based System

In order to carry out test I9T1.1 and test I9T2.1, an extra test driver is needed to simulate a generic end user that performs actions over the mobile application. Of course, this driver will not be replaced by any software component. Finally, as previously stated, there is a leaf component (Queue Manager) that in order to be properly tested needs not only a driver but also a stub. This stub will mock the Communication Manager component in tests I5T1.1 and I5T1.2.

6. Used Tools

6.1. Used Tools

The tools used to create the Test Plan document are:

- Microsoft Office Word 2011: to redact and to format this document.

For redacting and writing this document we have spent **15 hours** per person.