myTaxiService
Integration Document

Jacopo Strada Luca Riva

January 21, 2016

Version 1.0

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to guide through the integration testing of the various components described in the DD. In order to do this diagrams and tables are provided, so that it is possible to understand the approach selected for this kind of testing.

1.2 Scope

As properly described in the RASD this system was conceived in order to create an efficient platform to easily call or reserve a taxi, improving the quality of service for the clients and simplifying the job of the drivers. The scope of the tests in this document are meant to guarantee the right functioning of the system's components, assuming that the more specific unit tests have already been made, as reported in the entry criteria.

1.3 Glossary

1.3.1 Definitions

Client / Passenger / User : Is a person who signed up for this service and their interest is to call a taxi or reserve a ride.

Taxi Driver : Is a person who drives a taxi and would like to be called or reserved for a ride through this service.

Driver: A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system

Stub: A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.

1.3.2 Acronyms

GPS: Global Positioning System

DD: Design Document

RASD: Requirements and Specification Document

1.3.3 Abbreviations

IT*n*: Integration Test number n

1.4 Reference Documents

- *myTaxiDriver* Specification Document
- *myTaxiDriver* Requirements and Specification Document
- *myTaxiDriver* Integration Tests Assignment

2 Integration Strategy

2.1 Entry Criteria

These are the documents that must be delivered in order to perform the integration test: *RASD, DD*. In addition to these documents it is also required that all the components have been delivered and their respective unit tests have already been performed and passed successfully.

In order to proceed with the integration test is also important to have all the drivers that was used in the unit test because they can be reused in order to perform the necessary calls on the appropriate component that is now, step by step, connected with the others component of the system.

Another important thing is the interaction with the externals component that are: *Google Maps* and *Driver License DB*. These interactions are supposed to be already tested in the unit test because are very simple and really related only with the interested component.

2.2 Elements to be Integrated

The elements which require integration are:

- myTaxiService account manager
- myTaxiService DB
- PositionUtilities
- NotificationManager
- QueueManager
- CallManager
- ReservationManager
- myTaxiService MobileApp myTaxiService WebSite

The details about these elements and their interaction can be found in the design document.

2.3 Integration Test Strategy

The adopted strategy for this integration tests is almost always the *bottom-up* one. This approach was opted because it is possible to easily identify an atomic function for each component that will form the released software. As a consequence, testing the behaviour of every single component in the right order guarantees that the system as a whole will work properly.

Due to the complex interconnections present in the system, is not always possible to apply the pure *bottom-up* strategy that consist in creating only drivers. In fact, sometimes, in order to test only one interconnection at a time some stubs are needed to simulate the presence of a component whose interconnection wasn't already tested.

In order to keep the reading of the document as clearer as possible, for each integration test we have put a diagram representing the involved components and for each component a color is representative of the role of it in the specific integration test. Here is a small legend of the used colors:

Gray: Components that are under testing in the specific section.
Green: Components whose integration was alerady tested.
Blue: Stubs used for the test.
Red: Drivers used for the test.

2.4 Sequence of component

In this section are proposed two schemata which describe our components with their interconnections. For each connection an integration test is needed and marked with a code. The number present in the code identify the test and the order in which they must be performed.

As you can see the big *Application* component (highlighted in yellow) must be tested internally before starting with the highest level integration.

Regarding the two components Mobile Application and Web Application, here are always shown separately in order to be coherent with the DD but the explanation of tests is performed together referring to a *User Interface* component because the same tests must be done in the same way either for the site and for the app.

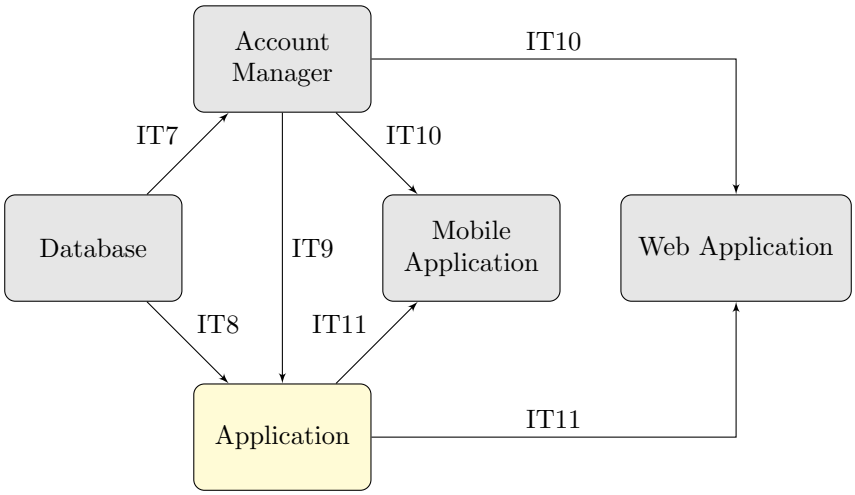

Figure 1: Integration schema between first level components

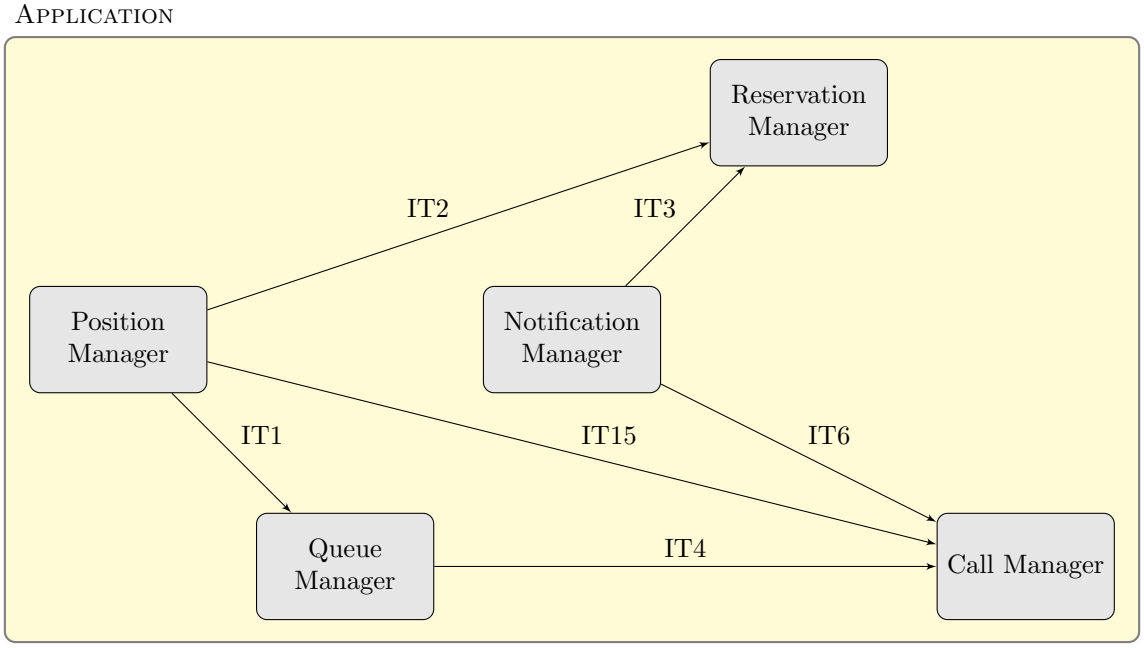

Figure 2: Integration schema between components belonging to application component

3 Individual Steps and Test Description

3.1 Integration Tests between Position Manager and Queue Manager

Figure 3: Integration schema of IT1 between Position Manager and Queue Manager

Test Identifier	IT1.1
Input Specification	GPS Coordinates
Output Specification	The reference to the zone in which the passed position is located.

Table 1: Integration Test between Position Manager and Queue Manager: Find Zone

Test Identifier	IT1.2
Input Specification	Driver identification
Output Specification	GPS Coordinates of their position

Table 2: Integration Test between Position Manager and Queue Manager: Get driver position

3.2 Integration Tests between Position Manager and Reservation Manager

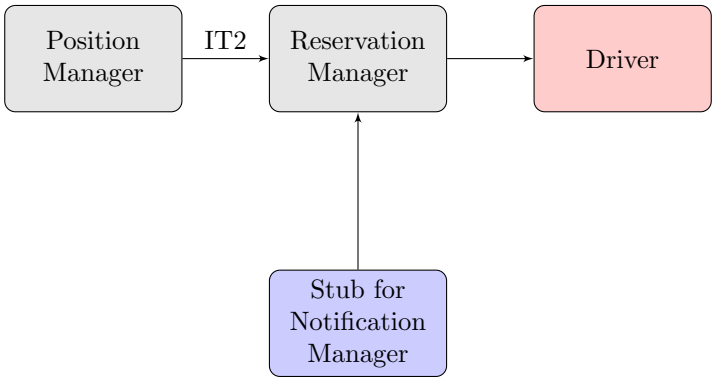

Figure 4: Integration schema of IT2 between Position Manager and Reservation Manager

Test Identifier	IT2
Input Specification	GPS Coordinates
Output Specification	The reference to the zone in which the passed position is located.

Table 3: Integration Test between Position Manager and Reservation Manager: Find Zone

3.3 Integration Tests between Notification Manager and Reservation Manager

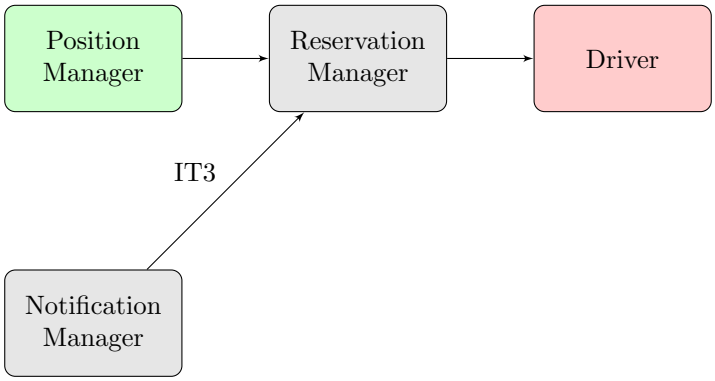

Figure 5: Integration schema of IT3 between Notification Manager and Reservation Manager

Test Identifier	IT3
Input Specification	list of email address and message
Output Specification	Notifications are registered by the notification manager

Table 4: Integration Test between Notification Manager and Reservation Manager: Send Notification

3.4 Integration Tests between Queue Manager and Call Manager

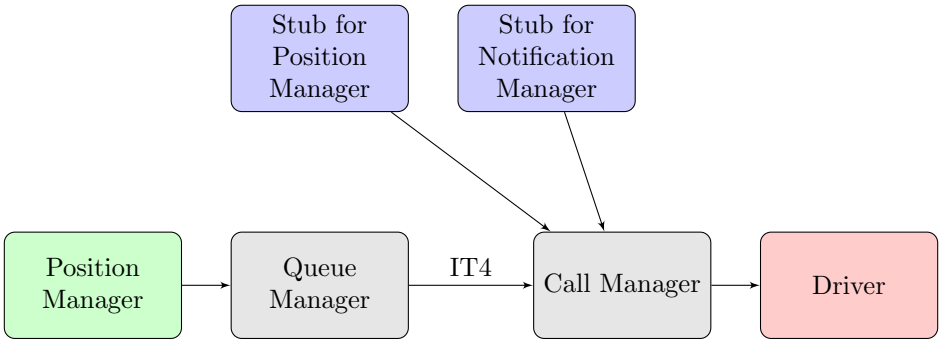

Figure 6: Integration schema of IT4 between Queue Manager and Call Manager

Test Identifier	IT4
Input Specification	Identifier of a zone
Output Specification	Return the drivers present in the specified zone

Table 5: Integration Test between Queue Manager and Call Manager: Get drivers present in a queue

3.5 Integration Tests between Position Manager and Call Manager

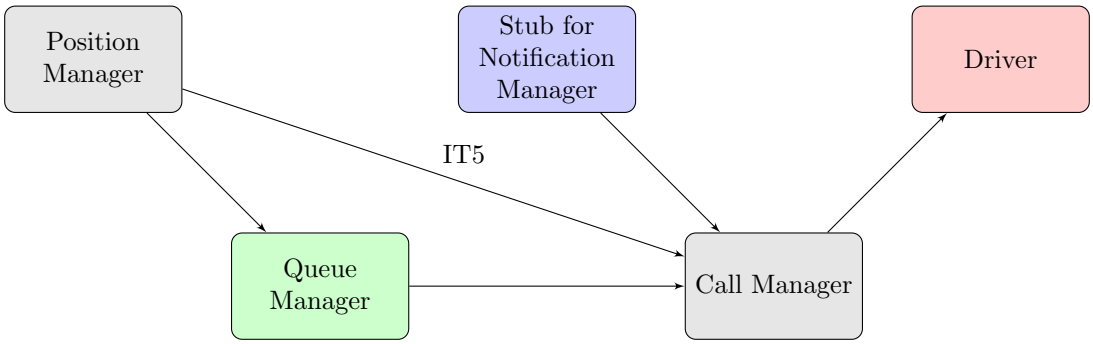

Figure 7: Integration schema of IT5 between Position Manager and Call Manager

Test Identifier	IT5
Input Specification	GPS Coordinates
Output Specification	The reference to the zone in which the passed position is located.

Table 6: Integration Test between Position Manager and Call Manager: Find Zone

3.6 Integration Tests between Notification Manager and Call Manager

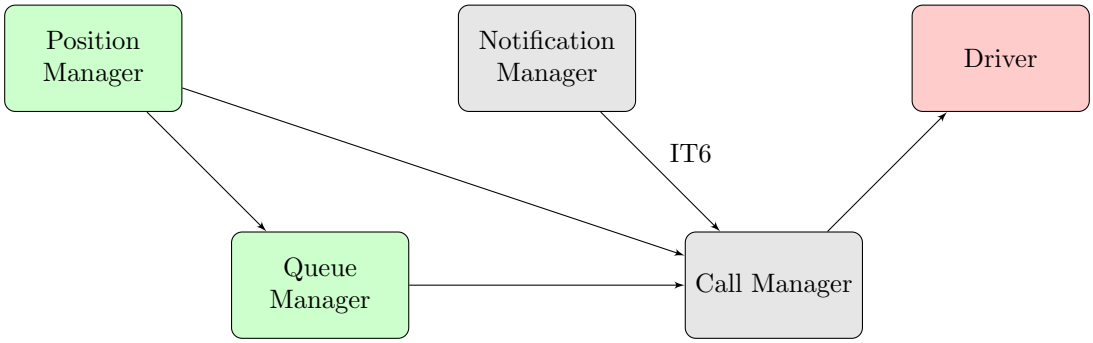

Figure 8: Integration schema of IT6 between Notification Manager and Call Manager

Test Identifier	IT6
Input Specification	list of email address and message
Output Specification	Notifications are registered by the notification manager

Table 7: Integration Test between Notification Manager and Call Manager: Send Notification

3.7 Integration Tests between Database and Account Manager

Figure 9: Integration schema of IT7 between Database and Account Manager

Test Identifier	IT7.1
Input Specification	Client data (email, name, surname, date of birth, phone, password) for creating a new account
Output Specification	A new client with the specified parameters was created and is present in the database

Table 8: Integration Test between Database and Account Manager: Creation of a new client

Test Identifier	IT7.2
Input Specification	Driver data (email, name, surname, license, phone, password) for creating a new account
Output Specification	A new driver with the specified parameters was created and is present in the database

Table 9: Integration Test between Database and Account Manager: Creation of a new driver

Test Identifier	IT7.3
Input Specification	A driver's or client's ID and their password
Output Specification	Accept the password if it is stored in the DB and relative to the user's ID

Table 10: Integration Test between Database and Account Manager: Password Check

Test Identifier	IT7.4
Input Specification	The e-mail of an user
Output Specification	The user registered in the database with the input e-mail or a message to notice that there is no registered user with that e-mail

Table 11: Integration Test between Database and Account Manager: Find user by their e-mail

Test Identifier	IT7.5
Input Specification	The e-mail of a driver
Output Specification	The state of the driver who has that e-mail, if there is no such driver an exception

Table 12: Integration Test between Database and Account Manager: Get the state of a driver

Test Identifier	IT7.6
Input Specification	The e-mail of a driver and their state to write in the DB
Output Specification	The updated state of the driver with that e-mail is written in the DB, if there is no such driver an exception in raised

Table 13: Integration Test between Database and Account Manager: Update the state of a driver

3.8 Integration Tests between Data Base and Application

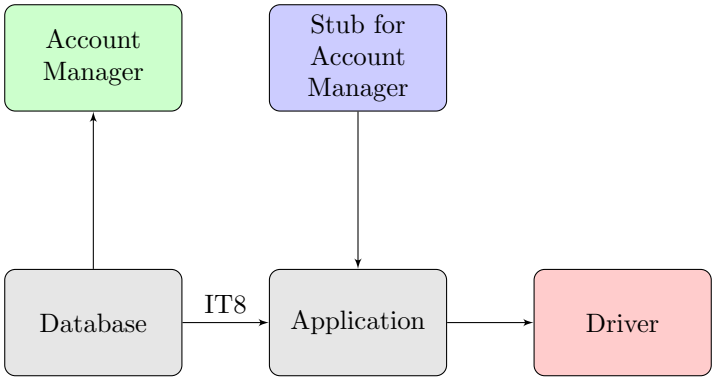

Figure 10: Integration schema of IT8 between Data Base and Application

Test Identifier	IT8.1
Input Specification	Request of zones
Output Specification	All the zones in which the city is divided are returned

Table 14: Integration Test between Data Base and Application: Get city zones

Test Identifier	IT8.2
Input Specification	Ride Reservation data (origin, destination, date, time)
Output Specification	The ride is stored in the database

Table 15: Integration Test between Data Base and Application: Save a ride reservation

Test Identifier	IT8.3
Input Specification	Ride Reservation ID and the driver who accepted the ride
Output Specification	The ride is set as accepted with the reference to the driver. If the ride isn't present in the database an error is displayed.

Table 16: Integration Test between Data Base and Application: Set a ride reservation as accepted

Test Identifier	IT8.4
Input Specification	Ride Reservation ID
Output Specification	The ride is deleted. If the ride isn't present yet in the database an error is displayed.

Table 17: Integration Test between Data Base and Application: Delete a Reservation

3.9 Integration Tests between Account Manager and Application

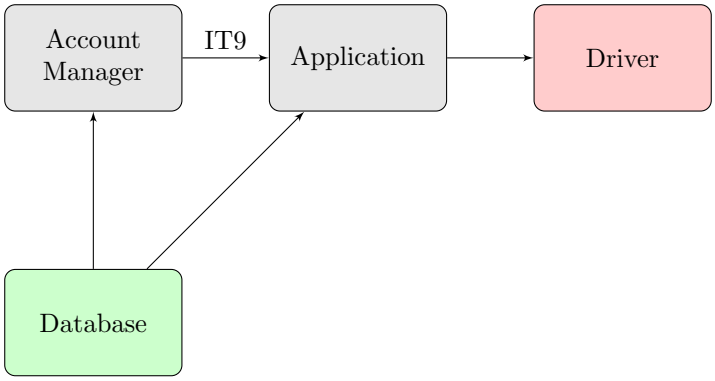

Figure 11: Integration schema of IT9 between Account Manager and Application

Test Identifier	IT9.1
Input Specification	email of a user (client or driver)
Output Specification	All the information relative to the passed user. If the user doesn't exists, an error is returned.

Table 18: Integration Test between Account Manager and Application: User info

Test Identifier	IT9.2
Input Specification	driver email and his state to be updated
Output Specification	the driver state is updated

Table 19: Integration Test between Account Manager and Application: Update driver state

Test Identifier	IT9.3
Input Specification	driver identifier
Output Specification	the driver is inserted in the right queue

Table 20: Integration Test between Account Manager and Application: Insert Driver in Queue

Test Identifier	IT9.4
Input Specification	driver identifier
Output Specification	Driver is no longer present the queue

Table 21: Integration Test between Account Manager and Application: Remove Driver from queue

3.10 Integration Tests between Account Manager and User Interface

In order to successfully test this part (and the following one) of the system the most efficient way is to find a group of drivers and possible clients who could test the main functionalities, in fact, having used a bottom up approach, the components which interact with the GUI should have already been properly tested at this point. At the same time it is important to collect opinions from a group of user in order to fix any eventual bug and to improve the application in terms of simplicity of usage and the quality of graphic design.

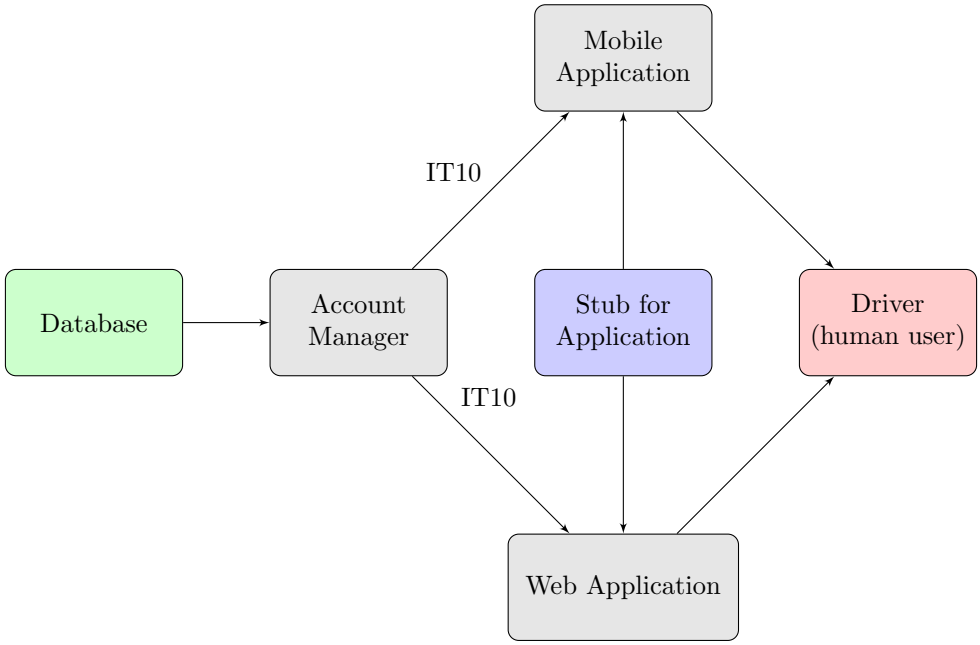

Figure 12: Integration schema of IT10 between Account Manager and User Interface

Test Identifier	IT10.1
Input Specification	A mail and a password
Output Specification	Acceptance or rejection of the request of login depending on the correctness of the credentials provided

Table 22: Integration Test between Account Manager and User Interface: Login of an user

Test Identifier	IT10.2
Input Specification	User data (email, first name, last name, telephone number, password) of the client who would like to be registered
Output Specification	The client account is successfully created and can be used to log in if (and only if) all the data are correct.

Table 23: Integration Test between Account Manager and User Interface: Client registration

Test Identifier	IT10.3
Input Specification	User data (email, first name, last name, telephone number, license, password) of the driver who would like to be registered
Output Specification	The driver account is successfully created and can be used to log in if (and only if) all the data are correct and the license is valid.

Table 24: Integration Test between Account Manager and User Interface: Driver registration

Test Identifier	IT10.4
Input Specification	email of a user (client or driver)
Output Specification	All the information relative to the passed user. If the user doesn't exists, an error is returned.

Table 25: Integration Test between Account Manager and User Interface: User info

Test Identifier	IT10.5
Input Specification	driver email and his state to be updated
Output Specification	the driver state is updated

Table 26: Integration Test between Account Manager and User Interface: Update driver state

3.11 Integration Tests between Application and User Interface

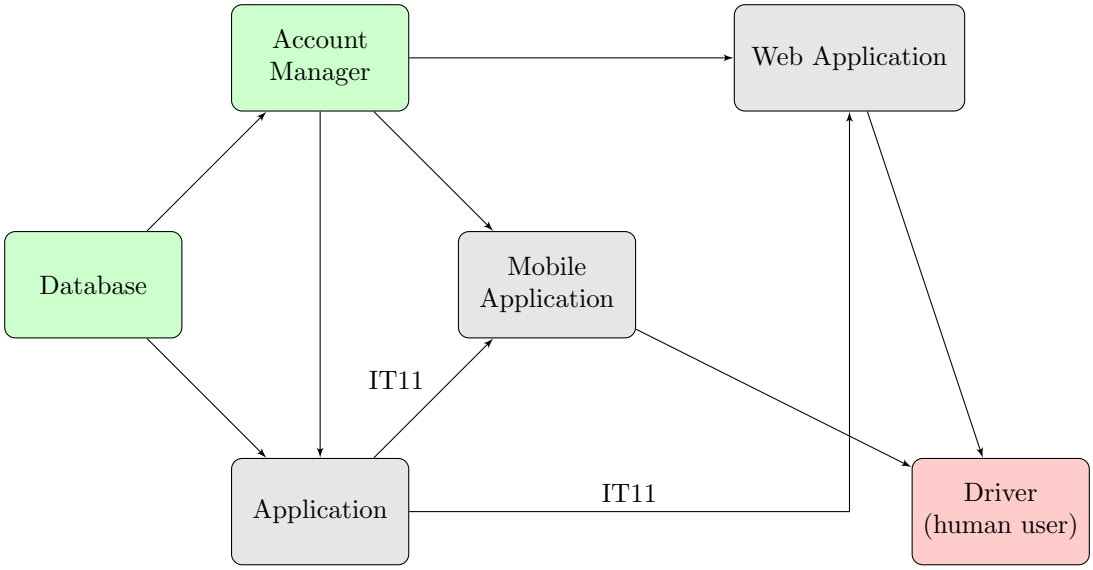

Figure 13: Integration schema of IT11 between Application and User Interface

Test Identifier	IT11.1
Input Specification	Client email address and their GPS coordinates
Output Specification	the call is submitted to the system

Table 27: Integration Test between Application and User Interface: Forward a call

Test Identifier	IT11.2
Input Specification	Call reference
Output Specification	all the details of the specific call are returned

Table 28: Integration Test between Application and User Interface: Show call details

Test Identifier	IT11.3
Input Specification	Client email, ride origin and destination, starting time
Output Specification	The reservation is correctly submitted to the system

Table 29: Integration Test between Application and User Interface: Create new Reservation

Test Identifier	IT11.4
Input Specification	Reservation reference
Output Specification	all the details of the specific reservation are returned

Table 30: Integration Test between Application and User Interface: Show reservation details

4 Tools and Test Equipment Required

In order to test the integration of our system there aren't a lot of useful tools and we've proceeded principally by manual tests.

A tool that can be used is *Mockito* which sound as a tool for unit tests but is important to simplify the creation of stubs which are used for the integration tests presented before.

In order to test the front-end of the system is important to have human people such as a team of taxi drivers and one of candidate users. In this way the system is checked in all its parts and also a feedback from the user is received.

5 Program Stubs and Test Data Required

Despite we used a bottom-up approach, the integration of some components cannot be correctly tested without the use of some stubs.

For this reasons a stub representing each of the following components is needed for our integration test:

- Notification Manager
- Position Manager
- Account Manager
- Application

The application that we are going to test is strictly related with a database and for this reason it must be initialized with some data in order to perform meaningful test. In order to do this is important to provide a set of users, both *Clients* and *Drivers* and a set of *Zones*. Having these data we can perform all the test in this simulated but real situation.

6 Appendix

6.1 Software and Tools used

ShareLatex: This web application was used to redact this document in a collaborative way. (`https://it.sharelatex.com/`)

6.2 Hours of Work

We spent approximately the following amount of hours to redact this document:

Riva Luca: 10

Strada Jacopo: 10