Integration Test Plan

myTaxiService

AA 2015/2016

POLITECNICO
MILANO 1863

Carlo Broggi

Nicola Ghio

Summary

1 Introduction

1.1 Revision History

Document Version Number	Date
1.0	15TH January 2016
1.1	16TH January 2016
1.2	17TH January 2016
2.0	18TH January 2016
2.1	21TH January 2016

1.2 Purpose and Scope

This document describes how to held the testing procedure while integrating all the created objects. The purpose of this document is to test the interfaces between the components of our system that has been already described in the DD (refer to page 7 of the DD). Every team member who cooperates during the components integration must carefully read this document.

As already explained in the RASD and the DD, we want to build a system that must provide a fair distribution of the work between all taxi drivers located in the city and a quick and reliable service for all the customers that want to reserve or request a taxi. We want provide an intuitive way of interaction for the users (taxi driver or customer) and also to give all the information they need as soon as they need it . Now we will integrate all the components that our system needs to work, and we will test that all functionalities work correctly to provide the best user experience possible.

1.3 List of Definitions and Abbreviations

Here is a list of all the terms we will use into this document

- *RASD: Requirement Analysis Specification Document*
- *All the terms already included in the RASD*
- *DD: Design Document*
- *All the terms already included in the DD*

1.4 List of Reference Documents

In document we could refer sometimes both to the RASD and to the DD. Reading this two document is suggested to fully know how the system will work and how all the component will interact between them.

During testing could be useful the Mockito and JMeter documentation.

We also referred to the IEEE standard to correctly structure this document.

2 Integration Strategy

2.1 Entry Criteria

All the module that have been developed must pass a phase of unit testing before being integrated. This can avoid in a certain way the malfunction of the single component of the system. After all components has been verified must start the integration procedure where we well test that the interfacing between all the components is correct

2.2 Elements to be Integrated

We want to integrate the following main components in our system:

- *Ride Manager*
- *Database Controller*
- *Session Manager*
- *Administration Controller*
- *Customer Controller*
- *Driver Controller*
- *Administration View (Browser)*
- *Customer View (Browser/Application)*
- *Driver View (Application)*
- *Reserve Ride (API)*

2.3 Integration Testing Strategy

We chose to use a mixed strategy during the integration process because we want to increase the parallelisation of the process without arming the quality of the system. If any problem arises from one of the testing flow the other functionalities can also be tested because they are independent one from another.

We Identify the following subsystems:

- *Administrator View*
- *Customer View*
- *Driver View*
- *Core*

We decide to integrate every component In each subsystem as specified in detail below.

For the component in a subsystem which are dependent from other component in the same subsystem we decided to use a bottom up approach in order to be certain that they cooperate correctly.

For the Core we identified a sequence of integration tests that allows us to test independent component, therefore we will perform those test as specified below and then we will proceed with the integration of the whole subsystem.

We used this kind of approach in order to obtain the best result obtainable in our opinion.

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

Integration sequence of the Administrator View:

1. : Administrator Web View => Operations Handler

Integration sequence of the Customer View:

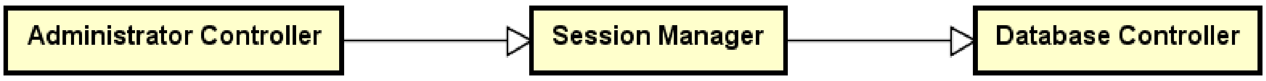
1. : Customer ApplicationWeb View => Customer Notification Handler
2. : Customer ApplicationWeb View => Customer Ride Handler
3. : Part(2)=> Customer Location GPS

Integration sequence of the Driver View:

1. : Driver Application View => Driver Notification Handler
2. : Driver Application View => Driver Ride Handler
3. : Driver Location GPS

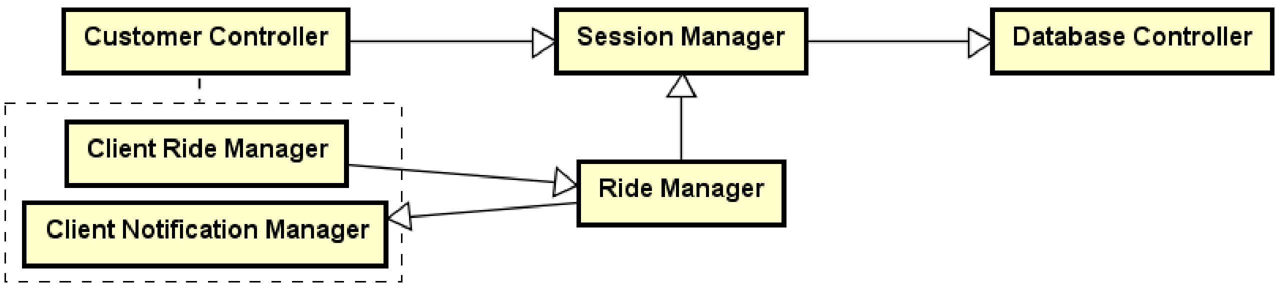
Integration sequence of the Core:

1. : Administrator Controller => Session manager
2. : Part (1) => Database Controller

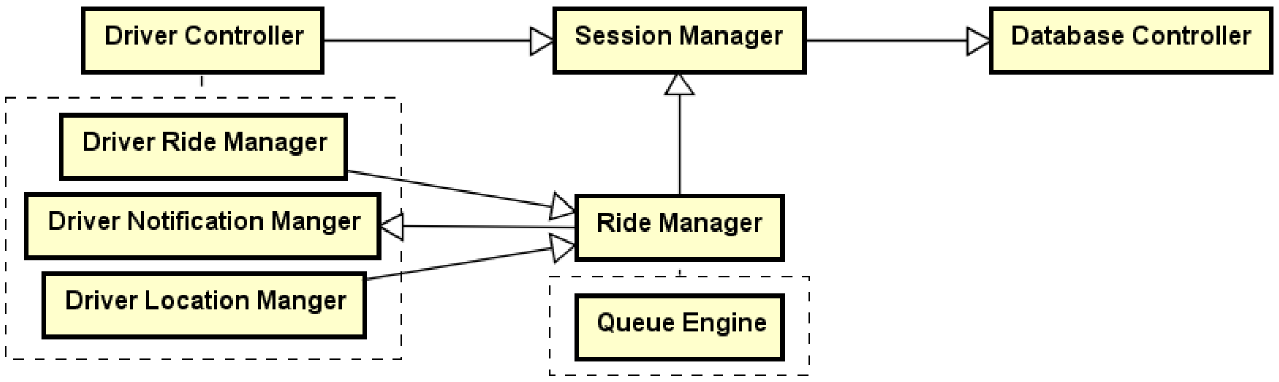


3. : Customer Controller => Session Manager
4. : Part(3) => Database Controller

5. : Client Ride Manager => Ride Manager
6. : Part(5) => Client Notification Manager
7. : Part(6) => Session Manager



8.	:	*Driver Controller => Session Manager*
9.	:	*Part(8) => Database Controller*
10.	:	*Driver Ride Manager, Driver Location Manager => Ride Manager, Queue Engine*
11.	:	*Part(10) => Driver Notification Manager*
12.	:	*Part(11) => Session manager*



13.	:	*API => Customer Controller*
14.	:	*Whole Core*

2.4.2 Subsystem Integration Sequence

We can also analyze in a better way the three subsystem regarding the views.

1.	:	*Core => Administrator View*
2.	:	*Part(1) => Driver View*
3.	:	*Part(2) => Customer View*

3 Individual Steps and Test Description

In this section we are going to explain in which test case we think are necessary to integrate our system. For each of them we give a brief description and we specify if they must take place after another test succeded or not.

3.1 Administrator

Test case Identifier	*Adm-T1*
Tested Items	*Administrator Web View – Operations Handler*
Input specification	*Simulate any administrator request by GUI*
Output Specifications	*Request are elaborated and ready to be sent and responses are displayed correctly on the screen*
Environmental Needs	*A stub for Administrator Controller*
Tests Needed	*None*

Purpose*: All the functionalities provided in the RASD document for the Administrator should be implemented and working correctly.*

3.2 Customer

Test case Identifier	*Cus-T1*
Tested Items	*Customer ApplicationWeb View – Customer Ride Handler*
Input specification	*Simulate any request made by a customer by GUI*
Output Specifications	*Request is elaborated and ready to be sent*
Environmental Needs	*A stub for Client Ride Manager and Customer Location GPS*
Tests Needed	*None*

Test case Identifier	*Cus-T2*
Tested Items	*Customer ApplicationWeb View – Customer Ride Handler- Customer Location GPS*
Input specification	*Simulate any "Request" request made by a customer by GUI*
Output Specifications	*The GPS location of the customer is correct*
Environmental Needs	*A stub for Client Ride Manager*
Tests Needed	*Cus-T1*

Test case Identifier	*Cus-T3*
Tested Items	*Customer ApplicationWeb View – Customer Notification Handler*
Input specification	*Simulate any possible notification sent by the Core to the Customer*
Output Specifications	*Notifications are displayed correctly*
Environmental Needs	*A stub for Client Notification Manager*
Tests Needed	*None*

Purpose: *All the functionalities provided in the RASD document for the Customer should be implemented and working correctly.*

3.3 Driver

Test case Identifier	*Dri-T1*
Tested Items	*Driver Application View – Driver Notification Handler*
Input specification	*Simulate any possible notification sent by the Core to the Driver*
Output Specifications	*Notifications are displayed correctly*
Environmental Needs	*A stub for Driver Notification Manager*

Tests Needed	*None*

Test case Identifier	*Dri-T2*
Tested Items	*Driver Application View – Driver Ride Handler*
Input specification	*Simulate the functionalities provided to the driver*
Output Specifications	*Requests are elaborated and ready to be sent to the Core*
Environmental Needs	*A stub for Driver Ride Manager*
Tests Needed	*None*

Test case Identifier	*Dri-T3*
Tested Items	*Driver Location GPS*
Input specification	*Simulate the GPS functionality of drivers*
Output Specifications	*GPS location is sent periodically to the Core*
Environmental Needs	*A stub for DriverLocation Manager*
Tests Needed	*None*

Purpose: *All the functionalities provided in the RASD document for the Driver should be implemented and working correctly.*

3.4 Core

3.4.1 Administrator

Test case Identifier	*C-Adm-T1*
Tested Items	*Administrator Controller – Session Manager*

Input specification	*Simulate the login/logout of an administrator*
Output Specifications	*The Session Manager allows login, records the session of the administrator until the logout*
Environmental Needs	*A stub for Operations Handler a stub for Database Controller*
Tests Needed	*None*

Purpose: *This test aims to verify if the administrator can login/logout properly and if this information is recorded into the Database.*

Test case Identifier	*C-Adm-T2*
Tested Items	*Administrator Controller – Database Controller – Session Manager*
Input specification	*Simulate a typical session of the Administrator.*
Output Specifications	*Requests are properly elaborated and the response given is correct*
Environmental Needs	*A stub for Operations Handler*
Tests Needed	*C-Adm-T1*

Purpose: *This test aims to verify if, after login, administrator's requests are executed correctly and if the Database Controller provides the correct response. Moreover we want to test if every non-query action performed by the administrator is stored into the Database.*

3.4.2 Customer

Test case Identifier	*C-Cus-T1*
Tested Items	*Customer Controller– Session Manager*
Input specification	*Simulate the login/logout of a Customer*
Output Specifications	*The Session Manager allows login, records the session of the customer until the logout*

Environmental Needs	*A stub for Customer View a stub for Database Controller*
Tests Needed	*None*

Purpose: *This test aims to verify if the customer can login/logout properly and if this information is recorded into the Database.*

Test case Identifier	*C-Cus-T2*
Tested Items	*Customer Controller– Session Manager-Database Controller*
Input specification	*Simulate the login/logout of a Customer*
Output Specifications	*The session manager correctly stores and retrieves data from the database.*
Environmental Needs	*A stub for Customer View a*
Tests Needed	*C-Cus-T1*

Test case Identifier	*C-Cus-T3*
Tested Items	*Client Ride Manager – Ride Manager*
Input specification	*Simulate a number of ride/reservation request from Customers*
Output Specifications	*Requests are properly elaborated and sent to Ride Manager*
Environmental Needs	*A stub for Customer Ride Handler*
Tests Needed	*None*

Purpose: *This test aims to verify whether a cluster of requests is handled properly or if the Core loses some information.*

Test case Identifier	*C-Cus-T4*
Tested Items	*ClientNotification Manager – Ride Manager – Client Ride Manager*

Input specification	*Simulate some request of ride/reservation from Customers*
Output Specifications	*Requests are properly elaborated and notifications are sent back*
Environmental Needs	*A stub for Customer Ride Handler - A stub for Customer Notification Handler*
Tests Needed	*C-Cus-T3*

Purpose: *This test aims to verify whether a cluster of requests is handled properly and if the responses are managed properly.*

Test case Identifier	*C-Cus-T5*
Tested Items	*ClientNotification Manager – Ride Manager – Client Ride Manager – Session Manager*
Input specification	*Simulate a whole session of a customer*
Output Specifications	*Requests are properly elaborated and notifications are sent back, sensible data are stored*
Environmental Needs	*A stub for Customer Ride Handler - A stub for Customer Notification Handler - A stub for Customer View - A stub for Database Controller*
Tests Needed	*C-Cus-T4*

Purpose: *This test aims to verify if all the part of the Core related to the customer work fine together, from the login to the response notification and if the data we supposed to store about customers are indeed stored.*

3.4.3 Driver

Test case Identifier	*C-Dri-T1*
Tested Items	*Driver Controller– Session Manager*
Input specification	*Simulate the login/logout of a Driver*
Output Specifications	*The Session Manager allows login, records the session of the customer until the logout*

Environmental Needs	*A stub for Driver View a stub for Database Controller*
Tests Needed	*None*

Purpose: *This test aims to verify if the driver can login/logout properly and if this information is recorded into the Database.*

Test case Identifier	*C-Dri-T2*
Tested Items	*Driver Controller– Session Manager-Database Controller*
Input specification	*Simulate the login/logout of a Driver with the database*
Output Specifications	*The Session Manager allows login, records the session of the customer until the logout*
Environmental Needs	*A stub for Driver View*
Tests Needed	*C-Dri-T1*

Test case Identifier	*C-Dri-T3*
Tested Items	*Driver Ride Manager – Ride Manager – Queue Engine – Driver Location Manager*
Input specification	*Simulate a number of queue request from Drivers*
Output Specifications	*Requests are properly elaborated and sent to Ride Manager*
Environmental Needs	*A stub for Driver Ride Handler- A stub for Driver Location GPS*
Tests Needed	*None*

Purpose: *This test aims to verify if the queue request are sent and received correctly. It also verifies if the Core reacts properly when the driver exit from a city zone while queuing in that zone.*

Test case Identifier	*C-Dri-T4*
Tested Items	*Driver Ride Manager – Ride Manager – Queue Engine – Driver Location Manager –*

	Driver Notification Manager
Input specification	*Simulate a number of queue and information request from Drivers*
Output Specifications	*Requests are properly elaborated and the Core responses properly*
Environmental Needs	*A stub for Driver Ride Handler - A stub for Driver Location GPS – A stub for Driver Notification Handler*
Tests Needed	*C-Dri-T3*

Purpose: *This test aims to verify if the request made by drivers to the Core (like information request about the queues into the city or a queue request) are properly elaborated and the response notification is correct.*

Test case Identifier	*C-Dri-T5*
Tested Items	*Driver Ride Manager – Ride Manager – Queue Engine – Driver Location Manager – Driver Notification Manager – Session Manager*
Input specification	*Simulate a whole session of a driver.*
Output Specifications	*Requests are properly elaborated and notifications are sent back, sensible data are stored.*
Environmental Needs	*A stub for Driver Ride Handler - A stub for Driver Location GPS – A stub for Driver Notification Handler*
Tests Needed	*C-Dri-T4*

Purpose: *This test aims to verify if all the part of the Core related to the driver work fine together, from the login to the response notification and if the data we supposed to store about drivers are indeed stored.*

3.4.4 API

Test case Identifier	*C-API-T1*
Tested Items	*Customer Controller*

Input specification	*Simulate an external request*
Output Specifications	*The request is elaborated properly.*
Environmental Needs	*A stub for Driver View a stub for Session manager and Ride manager*
Tests Needed	*C-Dri-T1*

3.4.5 Core complete Test

Test case Identifier	*C-All*
Tested Items	*All Core components*
Input specification	*Simulate any kind of interaction with the Core.*
Output Specifications	*All request are managed properly according to the RASD.*
Environmental Needs	*A stub for Driver View, Administrator View, Customer View*
Tests Needed	*All Core tests.*

3.5 Integrating Subsystems

In this part of the document we will explain how we plan to integrate the subsystem identified above in order to create the whole system.

3.5.1 Administrator – Core

Test case Identifier	*Subsystem-Integration-1*
Subsystems involved	*Administrator – Core*
Input specification	*Test how these subsystem work together, testing how administrator' functionalities are managed by the Core and how does it respond.*
Output Specifications	*Administrator can do anything properly and the Core outputs are correct.*

Environmental Needs	*Database filled with test data of Users in order to test the functionalities*
Subsystem Tests Needed	*None*

3.5.2 Administrator – Core – Driver

Test case Identifier	*Subsystem-Integration-2*
Subsystems involved	*Administrator – Core – Driver*
Input specification	*Test how these subsystem work together, focusing on testing all the lifecycle of a driver, from the creation to logout*
Output Specifications	*Administrator can properly manage drivers as specified into the RASD, Drivers can do anything properly and the Core outputs are correct.*
Environmental Needs	*Database filled with test data of Customers in order to test the functionalities*
Subsystem Tests Needed	*Subsystem-Integration-1*

3.5.3 Administrator – Core – Driver - Customer

Test case Identifier	*Subsystem-Integration-3*
Subsystems involved	*Administrator – Core – Driver - Customer*
Input specification	*Test how these subsystem work together, focusing on testing all the lifecycle of a customer, from the registraation to logout.*
Output Specifications	*Administrator can properly manage drivers and customers as specified into the RASD, Customers can do anything properly and the Core outputs are correct.*
Environmental Needs	*None*
Subsystem Tests Needed	*Subsystem-Integration-2*

From this moment the system can be considered complete.

4 Tools and Test Equipment Required

As stated in the entry criteria, all module are subjected to a phase of JUnit and manual testing before being integrated. This can avoid possible problem during integration procedure caused for example by data input or output that mismatch than they should be.

After we verify as possible the correctness of the modules, we can start integrating all the different part of our software. During the integration phase we use Mockito and again the Junit to test that our system being integrated part by part is working correctly in the way we are expecting. In every case required in the point number 3 of the document we will write some code ant then we will run this code to be sure that our integration work is going on correctly

Once the integration part is successfully terminated, we will finally use Jmeter to test the entire system in an high load environment to know if it is strong enough to be stable under lots of users that log in and generate requests. So we can verify that ours system can guarantee the condition of usage stated in the previous documents, trying to avoid software downtime caused by overloaded modules.

5 Program Stubs and Test Data Required

This part is included into the table of every single test that are stated in the point number three of this document.

For redacting and writing this document we spent more or less 11 hours per person.