Software Engineering 2
"myTaxiService"

Test Plan Document Version 1.0

1/21/2016

Politecnico di Milano A.A. 2015-2016

Milica Jovanovic (mat. 835953);Pavle Vidanovic (mat. 854472)

Contents

1 Introduction

1.1 Revision History

Version	Data	Authors	Summary
0.1	13/01/2016	Pavle Vidanovic Milica Jovanovic	Initial Draft
1.0	20/01/2016	Pavle Vidanovic Milica Jovanovic	Final Version

1.2 Purpose and Scope

The purpose of this document is to define plan for testing, integration testing and verifying that system development during the project complies with the requirements of Requirement document and Design Document. This document also presents test results in order to determinate if the application meets predetermined requirements and functionalities.

The aim of this project is to develop and implement myTaxiService, an application similar to Uber, which makes the process of assigning an available taxi vehicle to possible passengers.

The developed system should allow new users to register. Users, once logged in, should be able to:

- request a taxi
- reserve a taxi
- cancel a ride
- check taxi availability around him
- receive a confirmation with information about the assigned vehicle and ETA once taxi is requested
- create/maintain user profile
- report a taxi driver

The developed system should allow new taxi drivers to register. Drivers, once logged in, should be able to:

- inform the system about their availability
- confirm/decline that they are going to take care of a certain call
- create/maintain taxi driver profile
- report a passenger

The system should keep information about new arrived requests, as well as the confirmed rides. A ride should have and id number, information about the passenger that requested the ride, as well as the code of the assigned vehicle and ETA. System should also keep information about taxi queues connected to particular zone of the city and ensure fair management of the queues. Developed system should keep information about the list of reservations made by passengers, such as id number of the reservation, information about the passenger that made the reservation and the time of reservation and time of the ride.

1.3 Definitions and Abbreviations

ETA	Estimated Time of Arrival, approximated time of arrival of taxi vehicle to destination
Reservation	Passenger request for a vehicle at least 2 hours before the ride
Request	Passenger filled form for immediate ride
Reservation Conformation	Notification sent to the user about the confirmed reservation
Ride Conformation	Notification sent to the user about the confirmed ride with information of the ride
Report	Short description of problem that user/driver stumped into
User	A person already registered and logged into the system
Guest	A person accessing a system that has either never registered or hasn't logged in yet. Guest has only two available options, to log in or to register for the first time
Taxi driver	A person already register and logged into the system as a driver
GPS	Global Positioning System
API	Application Programming Interface.
DD	Design Document
DB	Database
DBMS	Database Management System
RASD	Requirement Analysis and Specification Document
ITPD	Integration Test Plan Document

1.4 Reference Documents

- RASD - RASD myTaxiService - final v2.0
- DD - DD myTaxiService - final
- Specification Document: myTaxiService Project AA 2015-2016
- Assignments 1 and 2 (RASD and DD)
- Assignment 4 - integration test plan
- Integration Test Plan Example

1.5 Document Overview

The document is essentially structured in six parts:

- Chapter 1: Introduction, gives description of document and some basic information about the software

- Chapter 2: Integration Strategy, gives an overview of entry criteria for the integrating components and how the elements will be integrated as well as used testing strategy and sequences of component/function integration
- Chapter 3: Individual Steps and Test Description, description of type of tests for verifying elements defined in one step, verifying the results are as expected
- Chapter 4: Tool and Test Equipment Required, overview on tools and equipment used to support integration test
- Chapter 5: Program Stubs and Test Data Required, gives an overview of how the requirements defined in RASD map into the design elements defined in DD.
- Chapter 6: References

2 Integration Strategy

2.1 Entry Criteria

Functions that need to pass Unit testing are entry criteria for following components of myTaxiService System:

Component	Functions to be unit tested
Guest Manager	• signUp() • signIn()
User Manager	• makeRequest() • makeReservation() • report() • manageProfile() • checkTaxisAvailable() • checkReservation() • cancelRide()
TaxiDriver Manager	• confirmDeclineRide() • setAvailable() • manageProfile() • report() • cancelRide() • checkRides()
Admin Manager	• banUser() • viewReports() • signIn()
Scheduler	
Request Manager	• createRequest() • provideTaxi() • calculateETA() • sendConfirmation() • findZone() • findDriver() • rideProposal()
Reservation Manager	• createReservation() • findDriver() • findZone() • sendConfirmation() • reservationConfirmation() • rideProposal()
Zone Manager	• determineZone() //getZone • findAvailableDriver() • enqueDriver() • dequeueDriver() • peekDriverOnQueue()

2.2 Elements to be integrated

Figures 2.1, 2.2, 2.3, 2.4 and 2.5 show the components that form the myTaxiService system. These figures are derived from figure 2.2 called Component view in DD, chapter 2. The arrows represent the order of integration, i.e. integration testing.

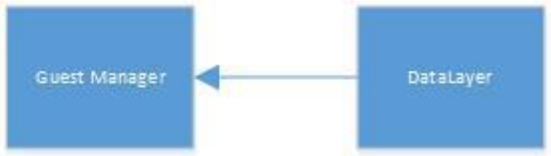

Figure 2.1 Guest Component

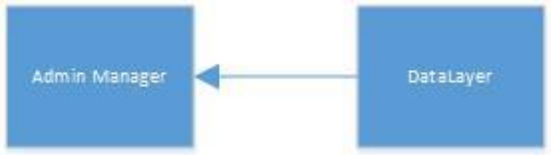

Figure 2.2 Admin Component

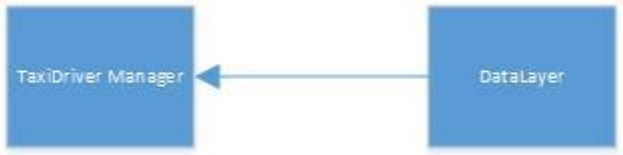

Figure 2.3 TaxiDriver Component – TaxiDriver application

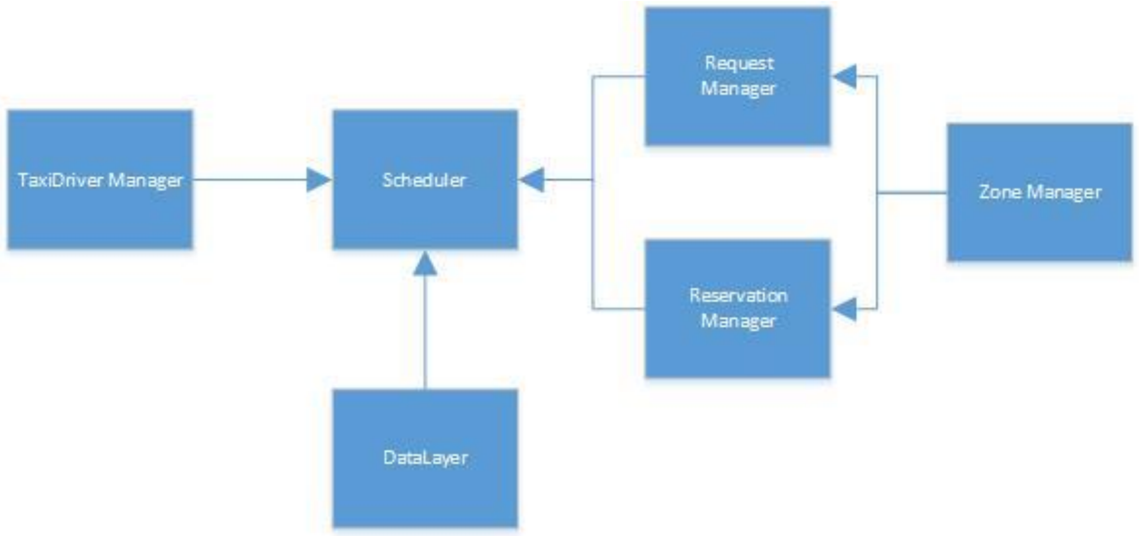

Figure 2.4 Scheduler Component

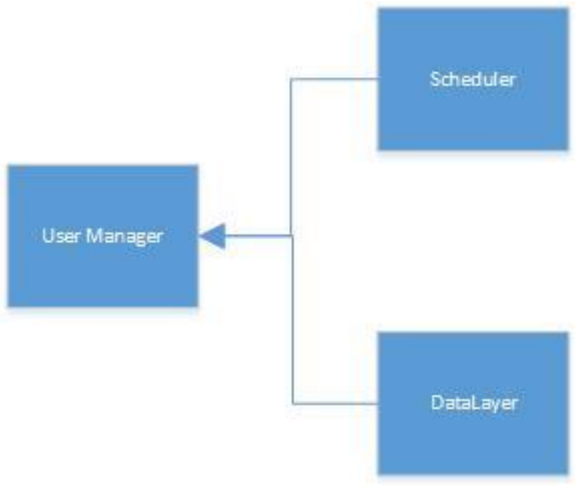

Figure 2.5 User Component – User application

2.3 Integration Testing Strategy

Items to be tested consist of integration of code modules developed for myTaxiService system. We have proposed as Integration Testing Strategy the bottom-up approach, starting from the lowest levels of the system (functions) that have passed the unit testing and that build up components. These components are integrated into bigger and complex components which in the end represent the myTaxiService system. One of the reasons for choosing bottom up approach is that it gives us a good overview of how far have we gone with the integration testing and as well to spot the problems in lower levels so they could be fixed before the system components are integrated at next level.

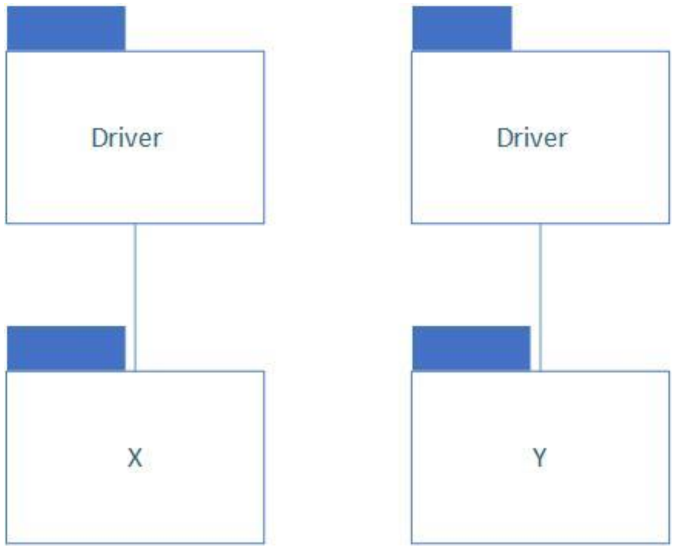

Figure 2.6

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

2.4.1.1 Integration Test of Guest Component

Integration Test of Guest Component relates to the Figure 2.1 of ITPD document.

ID	Integration Test
I1	DataLayer → Guest Manager

2.4.1.2 Integration Test of Admin Component

Integration Test of Admin Component relates to the Figure 2.2 of ITPD document.

ID	Integration Test
I2	DataLayer → Admin Manager

2.4.1.1 Integration Test of TaxiDriver Component

Integration Test of TaxiDriver Component relates to the Figure 2.3 of ITPD document.

ID	Integration Test
I3	DataLayer → TaxiDriver Manager

2.4.1.2 Integration Test of Scheduler Component

Integration Test of Scheduler Component relates to the Figure 2.4 of ITPD document.

ID	Integration Test
I4	Zone Manager → Request Manager, Reservation Manager
I5	Request Manager, Reservation Manager, TaxiDriver Manager, DataLayer → Scheduler

2.4.1.3 Integration Test of User Component

Integration Test of User Component relates to the Figure 2.5 of ITPD document.

ID	Integration Test
I6	DataLayer, Scheduler → User Manager

2.4.2 Subsystem Integration Sequence

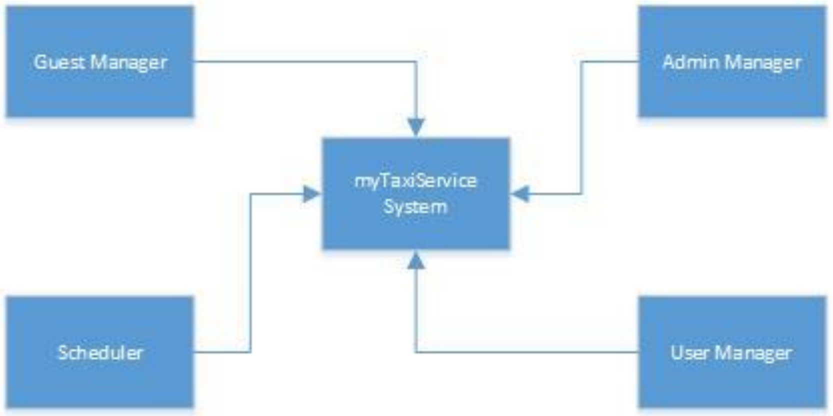

Figure 2.7 myTaxiService system – Subsystem integration

Integration Test of myTaxiService system from above integrated subsystem components relates to the Figure 2.6 of ITPD document.

ID	Integration Test
I7	Guest Manager, Admin Manager, User Manager, Scheduler → myTaxiSevice system

3 Individual Steps and Test Description

3.1 Test case specification

3.1.1 Integration test case I1

Test Case Identifier	**I1T1**
Test Item(s)	Data Layer → Guest Manager
Input Specification	Create typical DataLayer input as DB connection
Output Specification	Check if the correct functions are called in Guest Manager
Environmental Needs	N/A
Purpose	This test case checks whether the call made by guest (Guest Manager) work as expected: • signUp() • signIn()

3.1.2 Integration test case I2

Test Case Identifier	**I2T1**
Test Item(s)	Data Layer → Admin Manager
Input Specification	Create typical DataLayer input as DB connection
Output Specification	Check if the correct functions are called in Admin Manager
Environmental Needs	N/A
Purpose	This test case checks whether the call made by admin (Admin Manager) work as expected: • banUser() • viewReports() • signIn()

3.1.3 Integration test case I3

Test Case Identifier	**I3T1**
Test Item(s)	Data Layer → TaxiDriver Manager
Input Specification	Create typical DataLayer input as DB connection
Output Specification	Check if the correct functions are called in TaxiDriver Manager
Environmental Needs	N/A
Purpose	This test case checks whether the call made by driver (TaxiDriver Manager) work as expected: • manageProfile() • report()

	• checkRides() • setAvailable() • confirmDeclineRide() • cancelRide()

3.1.4 Integration test case I4

Test Case Identifier	**I4T1**
Test Item(s)	Zone Manager → Request Manager
Input Specification	Create typical Zone Manager input
Output Specification	Check if the correct functions are called in Request Manager
Environmental Needs	Request Driver
Purpose	This test case checks whether the call made by Request Manager work as expected: • findZone() • findDriver()

3.1.5 Integration test case I5

Test Case Identifier	**I5T1**
Test Item(s)	Zone Manager → Reservation Manager
Input Specification	Create typical Zone Manager input
Output Specification	Check if the correct functions are called in Reservation Manager
Environmental Needs	Reservation Driver
Purpose	This test case checks whether the call made by Reservation Manager work as expected: • findZone() • findDriver()

3.1.6 Integration test case I6

Test Case Identifier	**I6T1**
Test Item(s)	TaxiDriver Manager → Scheduler
Input Specification	Create typical TaxiDriver Manager input
Output Specification	Check if the correct functions are called in Scheduler Manager
Environmental Needs	I3 succeeded
Purpose	This test case checks whether the call made by Scheduler Manager work as expected:

	• rideProposal() • confirmReservation() • sendConfirmation()

Test Case Identifier	**I6T2**
Test Item(s)	Request Manager → Scheduler
Input Specification	Create typical Request Manager input
Output Specification	Check if the correct functions are called in Scheduler
Environmental Needs	I4 succeeded
Purpose	This test case checks whether the call made by Scheduler Manager work as expected: • provideTaxi() • calculateETA()

Test Case Identifier	**I6T3**
Test Item(s)	Reservation Manager → Scheduler
Input Specification	Create typical Reservation Manager input
Output Specification	Check if the correct functions are called in Scheduler
Environmental Needs	I5 succeeded
Purpose	This test case checks whether the call made by Scheduler Manager work as expected: • provideTaxi()

Test Case Identifier	**I6T4**
Test Item(s)	Data Layer → Scheduler
Input Specification	Create typical DataLayer input as DB connection
Output Specification	Check if the correct functions are called in Scheduler
Environmental Needs	N/A
Purpose	This test case checks whether the call made by Scheduler Manager work as expected: • createRequest() • createReservation()

3.1.7 Integration test case I7

Test Case Identifier	**I7T1**
Test Item(s)	Data Layer → User Manager
Input Specification	Create typical DataLayer input as DB connection
Output Specification	Check if the correct functions are called in User Manager
Environmental Needs	N/A
Purpose	This test case checks whether the call made by user (User Manager) work as expected: • report() • manageProfile() • checkTaxisAvailable() • checkReservation() • cancelRide()

Test Case Identifier	**I7T2**
Test Item(s)	Scheduler → User Manager
Input Specification	Create typical Scheduler input
Output Specification	Check if the correct functions are called in User Manager
Environmental Needs	I6 succeeded
Purpose	This test case checks whether the call made by user (User Manager) work as expected: • makeReservation() • makeRequest ()

3.2 Test procedures

3.2.1 Integration test procedure TP1

Test Procedure Identifier	**TP1**
Purpose	This test procedure verifies whether the Guest component: • Can handle guest input • Return correct information to the guest
Procedure Steps	Execute I1

3.2.2 Integration test procedure TP2

Test Procedure Identifier	**TP2**
Purpose	This test procedure verifies whether the Admin component: • Can handle admin input • Return correct information to the admin
Procedure Steps	Execute I2

3.2.3 Integration test procedure TP3

Test Procedure Identifier	**TP3**
Purpose	This test procedure verifies whether the TaxiDriver component: • Can handle TaxiDriver input • Return correct information to the TaxiDriver
Procedure Steps	Execute I3

3.2.4 Integration test procedure TP4

Test Procedure Identifier	**TP4**
Purpose	This test procedure verifies whether the Scheduler component: • Can handle TaxiDriver input • Can handle User input • Return correct information to the TaxiDriver • Return correct information to the User
Procedure Steps	Execute I6 after I3-I5

3.2.5 Integration test procedure TP5

Test Procedure Identifier	**TP5**
Purpose	This test procedure verifies whether the User component: • Can handle User input • Can output information to Scheduler • Return correct information to the User
Procedure Steps	Execute I7 after I6

4 Tools and Test Equipment Required

In order to have a more effective test, we decided to combine together the following tests:

- Manual Test
- Automatic Test

4.1 Manual

Team members dealing with user story both in client and on server side of the system respecting requirements written in Requirement Document. When a user story is finished, integration testing is done manually by the same team member who has made it. Then it is tested by other team member to ensure that all the requirements are met.

4.2 Automatic Test

After the manual test, team members will take care of writing code for the automatic test for each user story. User stories have more possible scenarios. Automating test allows us to save time in long term: it can be run in every moment and everybody can check reports. For automatic testing we use a combination of few parts of "Cucumber" and "Watir-Webdriver". "Ruby" is the programming language use for it.

5 Program Stubs and Test Data Required

Specifications of particular input data or component's stub/driver needed to perform the integration steps described in Chapter 3 are included in the list below:

• Test database: In order to perform some test cases, sample user data should be inserted into the database (DataLayer component) and made available for testing. These test data includes a reduced set of instances of all the entities

• External Google API stub: it is needed to replace the external GoogleMaps and GooglePlaces API system. This stub should provide sample data needed to the TaxiDriverManager component in order to correctly perform navigation, UserManager component to be able to pick his destination address and to be able to see available taxi vehicles near him on the map.

• External Gmail stub: it is needed to replace the external Gmail system. This stub should provide sample data needed to GuestManager when performing signUp procedure

• Drivers: generally, if some components may not be available yet for the integration test phase, they will be replaced with appropriate drivers (that take the part of those software component) in order to test the others

6 References

- Slides of the Software Engineering 2 course (Beep platform)
- Testing in Software Development, Martyn A. Ould, Charles Unwin, British Compute
- Official Cucumber website
- www.watirwebdriver.com – Watir-Webdriver