Release date 26-02-2015

Software Engineering 2: MyTaxiService

Integration Test Plan Document

V1.0

Dimitar Anastasovski, Marco Colombo

Contents

Chapter 1

Introduction

1.1 Purpose

This document will outline the system test plan that is in place and testing responsibilities. The purpose of the integration test plan is to describe the necessary tests to verify that all of the components of MyTaxiService are properly assembled. Integration testing ensures that the unit-tested modules interact correctly The purpose of this document is to define:

- The test scope, focus areas and objectives
- The test responsibilities
- The test strategy for the levels and types of test for this release
- The entry and exit criteria
- The basis of the test estimates
- Any risks, issues, assumptions and test dependencies
- The test schedule and major milestones
- The test deliverables

1.2 Scope

The main accent is to simplify and optimize the access of passengers to the system and to guarantee fair management of taxi queues. We will build flexible and user-friendly web application and a mobile application that will run on Android and IOS mobile phones. This application can be used by anyone who previously will be register on the registration page. After the registration is done the user will have a user name and password that should remember for furthermore usage of the system. The passenger can call a taxi after a successful logging on the application. After that he can call a taxi and he will be informed about the code of the incoming taxi, waiting time. On the other hand taxi drivers will have a mobile application where the major purpose will be to inform the system about their availability, confirmation of a certain call and global map navigation. City is divided into taxi zones that are uniquely associated with corresponding taxi queues for efficient usage of the system.

1.3 Definitions

Request	Passenger filled form for immediate ride
Reservation	Passengers can request for a vehicle at least 2 hours before the ride and can reserve his ride
User	Is a customer who already registered and logged into the system
Taxi driver	Is a person who legally drives taxi (with driver license and work license) already registered and logged into the system as a driver
System	Is the system that has to be designed
Taxi zone	Are the zones in which the city is divided in

1.4 Abbreviations

No abbreviations are been used in this document

1.5 Documents

Specification document	myTaxiService Project Document
RASD	*Requirements and Specification Document* version 1.0, November 2015
DD	*Design Document*, version 1.0, December 2015
ITPD Example	*Integration Test Plan Document Example* spinGRID, version 0.1.0, May 2006

Chapter 2

Integration strategy

2.1 Entry criteria

Registered Manager	SignUp() Sign In()

Licensed Taxi Drivers Manager	SignIn()
	CancelRide()
	ViewReservation()
	ViewMap()
	SetAvailability()

Unregistered User	SignUp()

Admin Manager	SignIn()
	ViewReports()
	BanUsers()

ETA Manager	ETA()
	ETAPrice()

Zone Manager	DetermineZone()
	AssignToAvailableDrivers()

Queues Manager	EnqueueDriver()
	DequeueDriver()

Scheduler Manager	CreateRequest()
	FirstAvailableTaxiInQueue()
	DetermineZone()
	SendConfirmation()

Reservation Manager	createReservation()
	FirstAvailableTaxiInQueue()
	DetermineZone()
	SendConfirmation()

2.2 Integrated elements

The following sections will describe the test cases that will be performed based on Component view diagram from DD-document . These test cases will be identified by the component that they test. You can see the figures below:

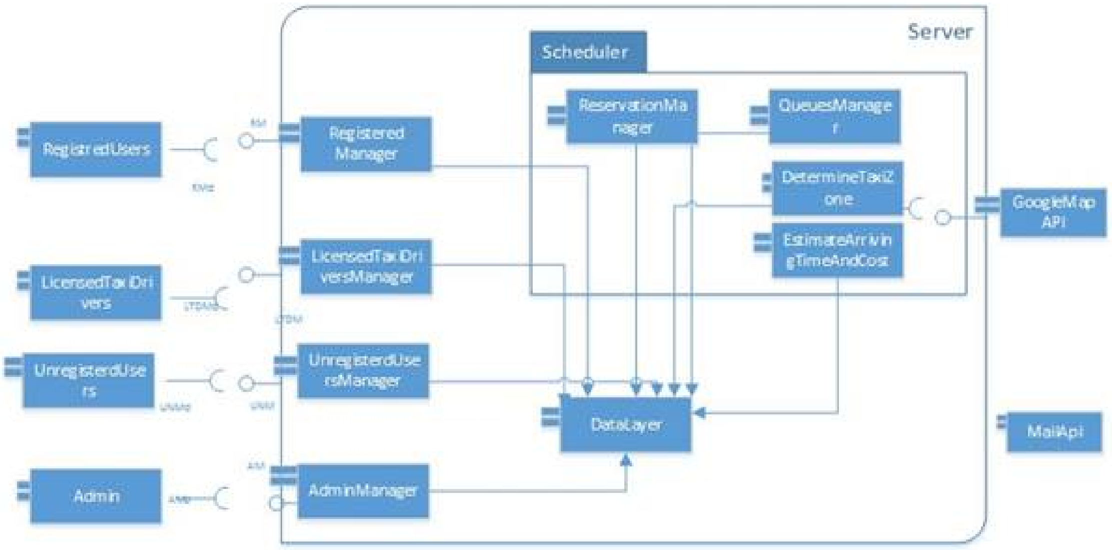


Figure 1: Component View

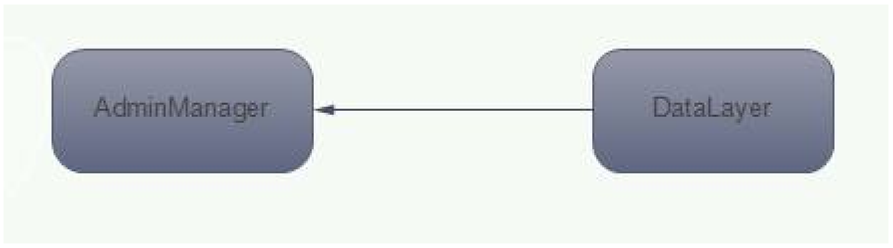


Figure 2: Component AdminManager

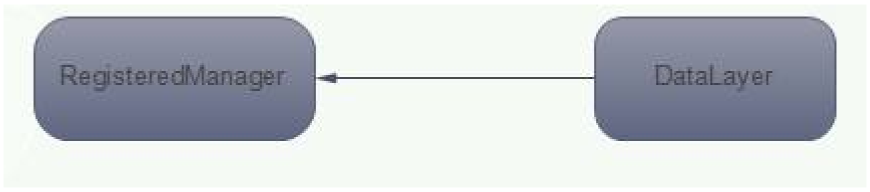


Figure 3: Component RegisteredManager

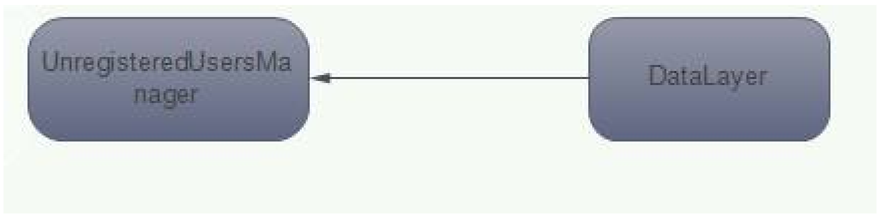


Figure 4: Component UnregisteredUserManager

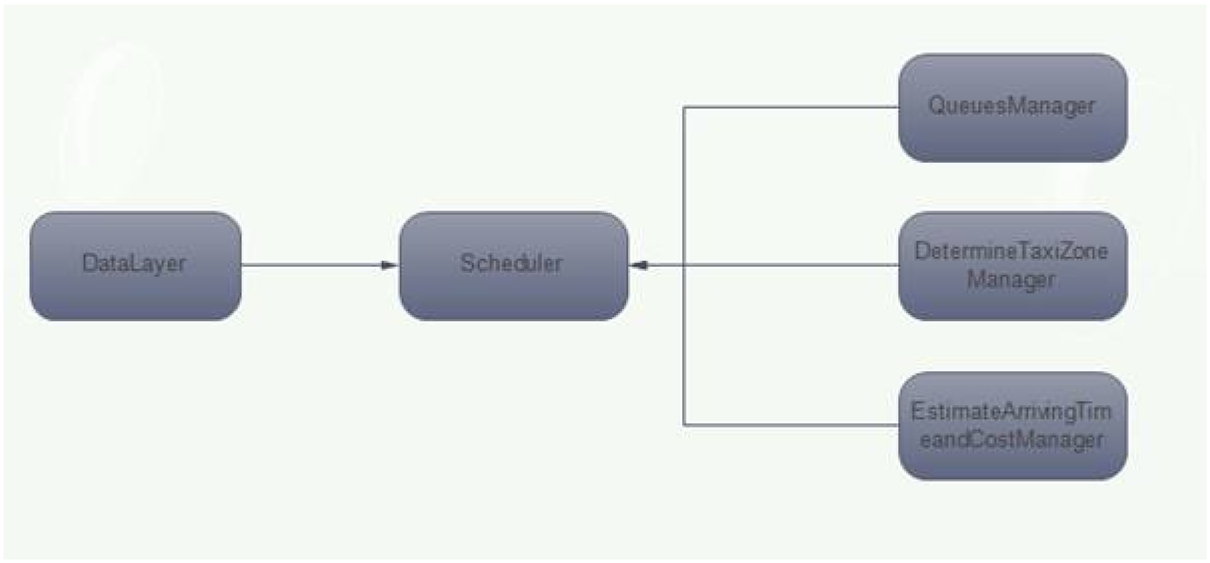


Figure 5: Component Scheduler

2.3 Integration testing strategy

For testing we choose a bottom-up strategy. This means that we began from the bottom levels to the top. We choose this strategy because we don't need stub and we can test all single parts before connect them together and obtain the whole project. The bottom-up approach implies proactive team input in the project executing process. Team members are invited to participate in every step of the management process. The decision on a course of action is taken by the whole team.

Chapter 3

Test specifications

3.1 Test case specifications

3.1.1 Integration test case I1T1

Test Case Identifier	I1T1
Test Item	Data Layer → RegisteredManager
Input Specification	Create typical data layer input
Output Specification	Check if the correct functions are called in RegisteredManager

3.1.2 Integration test case I2T1

Test Case Identifier	I2T1
Test Item	Data Layer → LicensedTaxiDriverManager
Input Specification	Create typical data layer input
Output Specification	Check if the correct functions are called in LicensedTaxiDriverManager

3.1.3 Integration test case I3T1

Test Case Identifier	I3T1
Test Item	Data Layer → AdminManager
Input Specification	Create typical data layer input
Output Specification	Check if the correct functions are called in AdminManager

3.1.4 Integration test case I4T1

Test Case Identifier	I4T1
Test Item	Data Layer → UnregisteredUserManager
Input Specification	Create typical data layer input
Output Specification	Check if the correct functions are called in UnregisteredUsersManager

3.1.5 Integration test case I5T1

Test Case Identifier	I5T1
Test Item	QueueManager → Scheduler
Input Specification	Create typical Queue input
Output Specification	Check if the correct functions are called in Scheduler

3.1.6 Integration test case I5T2

Test Case Identifier	I5T2
Test Item	DetermineTaxiZoneManager → Scheduler
Input Specification	Create typical Zone input
Output Specification	Check if the correct functions are called in Scheduler

3.1.7 Integration test case I5T3

Test Case Identifier	I5T3
Test Item	EstimateArrivingTimeandCostManager → Scheduler
Input Specification	Create typical ETA input
Output Specification	Check if the correct functions are called in Scheduler

3.1.8 Integration test case I5T4

Test Case Identifier	I5T4
Test Item	Data Layer $\rightarrow$ Scheduler
Input Specification	Create typical Data Layer input
Output Specification	Check if the correct functions are called in Scheduler

3.2 Test procedures

3.2.1 Test procedure T1

Test Procedure Identifier	T1
Test Items	TaxiDriver and Passenger
Input Specification	Create a new taxi reservation
Output Specification	Check if a new reservation is create

3.2.2 Test procedure T2

Test Procedure Identifier	T2
Test Items	Reservation and Notify
Input Specification	Send notification after reservation
Output Specification	Check if a notification is send to the passenger

3.2.3 Test procedure T3

Test Procedure Identifier	T3
Test Item	User
Input Specification	The user wants to log in into the system
Output Specification	Check if the user is logged in succesfully

3.2.4 Test procedure T4

Test Procedure Identifier	T4
Test Item	Passenger
Input Specification	The passemger wants to request a taxi
Output Specification	Check if the passenger has called a taxi succesfully

3.2.5 Test procedure T5

Test Procedure Identifier	T5
Test Item	Passenger
Input Specification	The passemger wants to delete a reservation
Output Specification	Check if the passenger has deleted the reservation

3.2.6 Test procedure T6

Test Procedure Identifier	T6
Test Item	Taxi driver
Input Specification	The taxi driver is available
Output Specification	Check if the taxi driver declined or accept the request

3.2.7 Test procedure T7

Test Procedure Identifier	T7
Test Items	Server and Users
Input Specification	The user wants to access to the application
Output Specification	Check if the user is already on the server

3.2.8 Test procedure T8

Test Procedure Identifier	T8
Test Items	Taxi and queue
Input Specification	The user request a taxi
Output Specification	Check if the taxi is available and remove from the queue

Chapter 4

Tools and test equipment

To make sure that this product will be on market without unexpected behavior and can be reliable two types of testing will be done before final commercial release.

Manual

The main goal of manual testing is to make sure that the application under test is defect free and software application is working as per the requirement specification document (RASD). This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type, tester takes over the role of end user and test the Software to identify any un-expected behavior or bug. Manual testing will be very useful while executing test cases first time, may or may not be powerful to catch the regression defects under frequently changing requirements. This test cost less than automated one.

Automatic

The objective of automated testing is to simplify as much of the testing effort as possible with a minimum set of scripts. Automated testing tools are capable of executing tests, reporting outcomes and comparing results with earlier test runs. Automation testing will be used when need to execute the set of test cases tests repeatedly. This type of testing will be very useful to catch regressions in a timely manner when the code is frequently changes also will be carried out simultaneously on different machine with different OS platform combination. For this project it's going to be used:

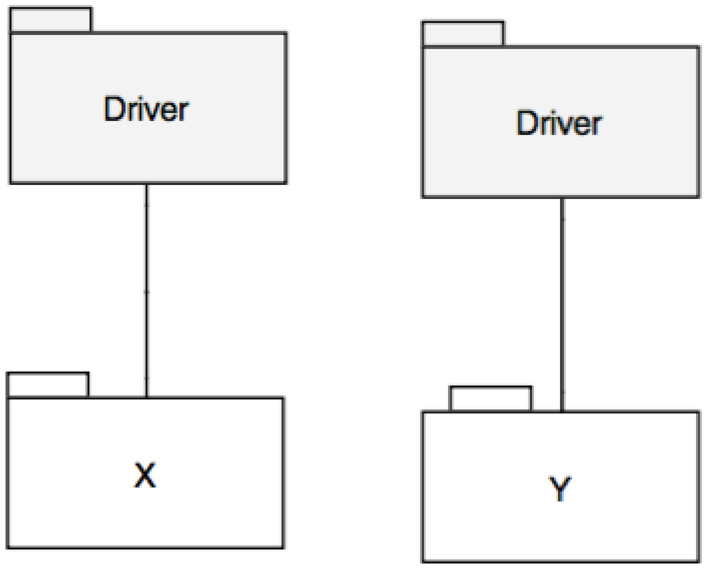
- For unit testing we will use JUnit. Is a simple unit testing framework to write repeatable tests in Java. JUnit tests do not require human judgment to interpret, and it is easy to run many of them at the same time.

- For the web application we will use Selenium. Selenium is a popular automated web testing tool and helps you automate web browsers across different platforms. Selenium has the support of some of the largest browser vendors who have taken steps to make Selenium a native part of their browser.
- For GUI we will use Ranorex. It allows us to automate our application and record user interactions and play them back to execute our tests. Ranorex is one of the more popular commercial tools to build and run automated GUI and web tests.

Chapter 5

Test data required

We don't need any stub because we choose a bottom-up approach, but we have to use drivers to test each module and also when we connect them together.



Chapter 6

References

6.1 Working hours

Dimitar Anastasovski: ∼ hours

Marco Colombo: ∼ hours