

Software Engineering 2 “myTaxiService”

Test Plan Document Version 0.1

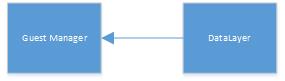
1/13/2016

Politecnico di Milano A.A. 2015-2016

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

Contents

[1 Introduction 4](#_Toc440541568)



[1.1 Revision History 4](#_Toc440541569)

[1.2 Purpose and Scope 4](#_Toc440541570)

[1.3 Definitions and Abbreviations 4](#_Toc440541571)

[1.4 Reference Documents 5](#_Toc440541572)

[1.5 Document Overview 5](#_Toc440541573)

[2 Integration Strategy 6](#_Toc440541574)

[2.1 Entry Criteria 6](#_Toc440541575)

[2.2 Elements to be integrated 7](#_Toc440541576)

[2.3 Integration Testing Strategy 7](#_Toc440541577)

[2.4 Sequence of Component/Function Integration 7](#_Toc440541578)

[2.4.1 SignUp 7](#_Toc440541579)

[2.4.2 Request 8](#_Toc440541580)

[2.4.3 Reservation 8](#_Toc440541581)

[2.4.4 Confirm/Decline a ride 9](#_Toc440541582)

[2.4.5 Ban User 9](#_Toc440541583)

[2.5 Component interfaces 10](#_Toc440541584)

[2.5.1 Guest Manager 10](#_Toc440541585)

[2.5.2 User Manager 10](#_Toc440541586)

[2.5.3 TaxiDriver Manager 11](#_Toc440541587)

[2.5.4 Admin Manager 11](#_Toc440541588)

[2.6 Selected architectural styles and patterns 12](#_Toc440541589)

[2.6.1 Architectural styles 12](#_Toc440541590)

[2.6.2 Patterns 15](#_Toc440541591)

[2.7 Other design decisions 17](#_Toc440541592)

[3 Algorithm design 18](#_Toc440541593)

[3.1 Request Algorithm 18](#_Toc440541594)

[3.2 Reserve Algorithm 19](#_Toc440541595)

[3.3 ConfirmDecline Algorithm 20](#_Toc440541596)

[3.4 SetAvailability Algorithm 21](#_Toc440541597)

[4 User interface design 22](#_Toc440541598)

[5 Requirement traceability 22](#_Toc440541599)

[6 References 22](#_Toc440541600)

# Introduction

## Revision History

## 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.

## 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 |

## 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

## 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

# Integration Strategy

## 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() |
| 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() |

## Elements to be integrated

Figures 2.1, 2.2, 2.3 and 2.4 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.

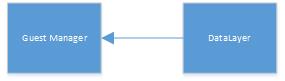


Figure 2.1 Guest Component

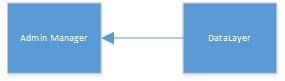


Figure 2.2 Admin Component

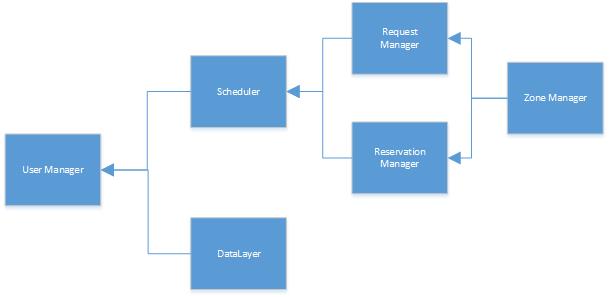


Figure 2.3 User Component – User application

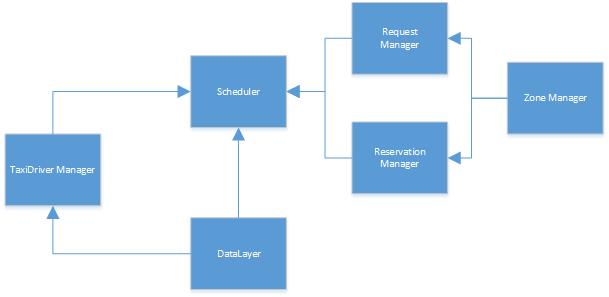


Figure 2.4 TaxiDriver Component – TaxiDriver application

## 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.

## Sequence of Component/Function Integration

### Software Integration Sequence

### 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 |

### 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 |

### Integration Test of User Component

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

|  |  |
| --- | --- |
| ID | Integration Test |
| I3 | Zone Manager Request Manager, Reservation Manager |
| I4 | Request Manager, Reservation Manager Scheduler |
| I5 | Scheduler, DataLayer User Manager |

### Integration Test of TaxiDriver Component

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

|  |  |
| --- | --- |
| ID | Integration Test |
| I6 | Zone Manager Request Manager, Reservation Manager |
| I7 | DataLayer TaxiDriver Manager |
| I8 | TaxiDriver Manager,Request Manager, Reservation Manager, DataLayer Scheduler |

### Subsystem Integration Sequence

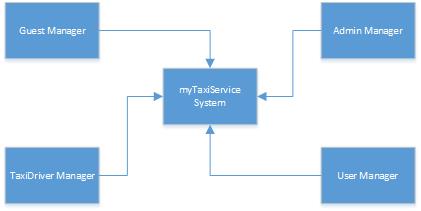


Figure 2.5 myTaxiService system – Subsystem integration

Integration Test of myTaxiService system from above integrated subsystem components relates to the Figure 2.5 of ITPD document.

|  |  |
| --- | --- |
| ID | Integration Test |
| I9 | Guest Manager, Admin Manager, User Manager, TaxiDriver Manager myTaxiSevice system |

# Individual Steps and Test Description

# Tools and Test Equipment Required

# Program Stubs and Test Data Required

Refer to the chapter 3.1 of RASD documents. In chapter 3.1 are shown user interfaces of myTaxiService system.

# References

DD References

* IEEE Design Document template
* MVC in a three-tier architecture, Peter Rawsthorne
* http://www.kony.com/resources/glossary/cross-platform-mobile-development
* https://msdn.microsoft.com/en-us/library/ee658117.aspx
* Singleton Design Pattern- Java Tech Guy must read!, Rudra Narayan Garnaik
* Software Design and Software Architecture, Politecnico di Milano
* Advanced Software Design and architectures: Architecture styles, Damian A. Tamburri