Taxi Service

Integration Test Documentation

Authors:

Liu Yuqi 14h

Xu Xiuli 14h

1. Introduction.. 3
1.1 Revision History..3
1.2 Purpose and Scope... 3
1.3 List of Definitions and Abbreviations..................................... 3
1.4 List of Reference Document.. 4
2. Integration strategy.. 4
2.1 Entry Criteria...4
2.2 Elements to be Integrated... 5
2.3 Integration Testing Strategy...6
2.4 Sequence of Component/Function Integration..................... 7
2.4.1 Software Integration Sequence.................................... 7
2.4.2 Subsystem Integration Sequence..................................8
3. Individual Steps and Test Description... 10
4. Tools and Test Equipment Required..11
5.Program and Test Equipment Required....................................... 11

1.Introduction

1.1 Revision History

First version (1.0) of the ITPD document.

1.2 Purpose and Scope

This document aims to describe, specify and analyze the integration test strategy for My Taxi Service, in terms of the components/classes to integrate and the typology of testing, while also providing a general schedule for the whole process; all is done accordingly to what was established in the previous assignments.

1.3 List of Definitions and Abbreviations

o Guest: it is an user that is not yet registered;

o Customer: it is an user that is registered and correspond to the passenger;

o Driver: it is an user that is registered and correspond to the taxi driver;

o External Developer: it is an user that is registered and can only require the API of the system;

o User: it could be a Guest, Customer, Driver or External Developer.

o Vehicle: correspond to the taxi car that is driven by a Driver;

o Request: it is a generic reservation made by a Customer;

o Booked Request: it is a specific Request, and it is made by the customer to book a taxi for a specific hour, origin and destination address;

o Zone: it indicates a specific area of the city that includes only one queue.

1.4 List of Reference Document

This is the list of the reference documents:

--Taxi service project specification;

--RASD document;

--DD document

2.Integration strategy

2.1 Entry Criteria

In order to start an integration test, two constraints must be satisfied: the major classes must be covered by, at least 60 percent of unit tests, while for the others a value of 30 percent is sufficient.

Major classes are: Userinterface, Activity, Action, Clientnetworkinterface, Servernetworkinterface, Controller, Ridesmanager, User, Ride,Sharedride, Taxiqueue.

2.2 Elements to be Integrated

ın our document, "element" is used as synonym of "class"; the following list describes the classes that need to undergo an integration test, in order to be sure that our application will behave correctly.

Ridesmanager$\longrightarrow$ Ride, Sharedrive	In order to store information about the activated rides
Integration Test: Ridesmanager Ridesmanager $\longrightarrow$Taxiqueue	In order to take information of available taxis in case of taxi request
Ridesmanager $\longrightarrow$ Controller	In order to exchange information about user’s requests

Integration Test: Controller

Controller $\longrightarrow$User	In order to create an ad-hoc Controller and to retrieve information about users
Controller$\longrightarrow$ Servernetworkinterface	In order to communicate with the corresponding client side

Integration Test: Servernetworkinterface

Servernetworkinterface $\longrightarrow$ Clientmessage	In order to read client’s messages
Servernetworkinterface $\longrightarrow$ Servermessage	In order to send messages to the client

Integration Test: Activity

Activity$\longrightarrow$ Action	In order to provide the allowed actions
Activity$\longrightarrow$Userintrface	In order to provide the set of items this class needs to show

Integration Test: Action

Action$\longrightarrow$Clientnetworkinterface	In order to send requests to the server

Integration Test: Userinterface

Userinterface$\longrightarrow$Clientnetworkinterface	In order to show the right Activity according to the server message

Integration Test: Clientnetworkinterface

Clientnetworkinterface⟶Clientmessage	In order to send messages to the server
Clientnetworkinterface⟶Servermessage	In order to read server's messages

2.3 Integration Testing Strategy

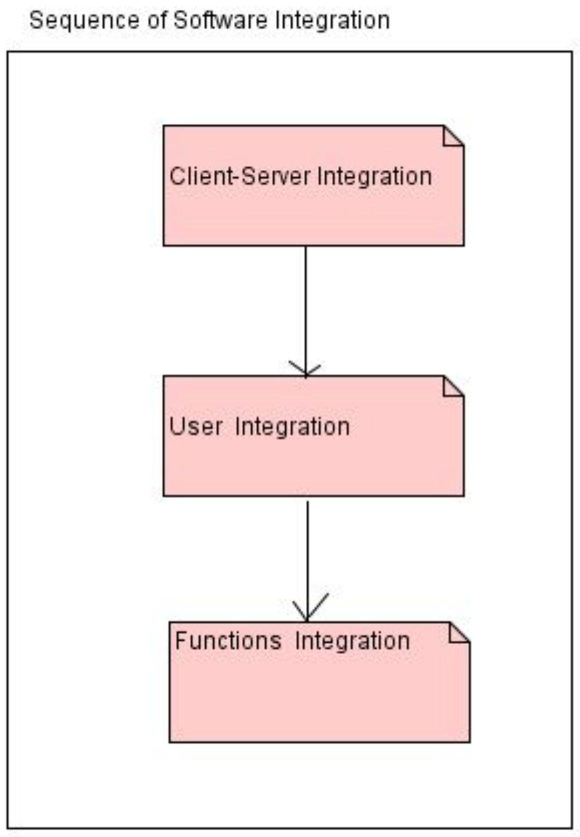
In this section we will explain how we planned the integration test in order to build, as soon as possible, a running application with few working features; this will allow us to promptly show our progress to the customer and also, in case of a delay in the development, to launch a working application, although with missing requirements. In order to reach our goal we decided to apply a bottom-up method during the integration test phase, and a top-down one for the unit tests.

The first working version of our application will include major classes; in this milestone, there are no users, but only a guest that has the possibility to access all the already implemented features.The second version will add multiple other users with the related constraints, as explained in the previous documents (see RASD and Design Document). From the second version onward, the application could be released, even if only few features are already implemented.The next versions will include other features that allow us to meet all missing requirements.

2.4 Sequence of Component/Function Integration

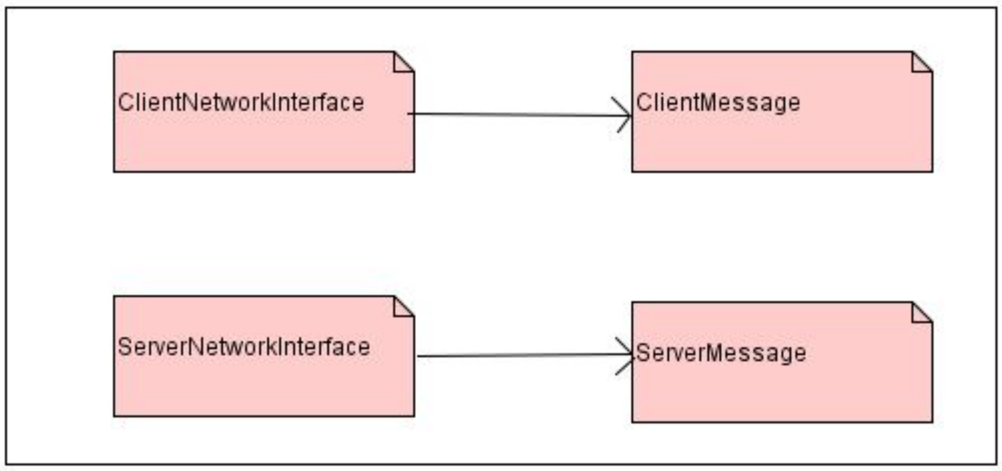
2.4.1 Software Integration Sequence

The software integration sequence is shown as follow:

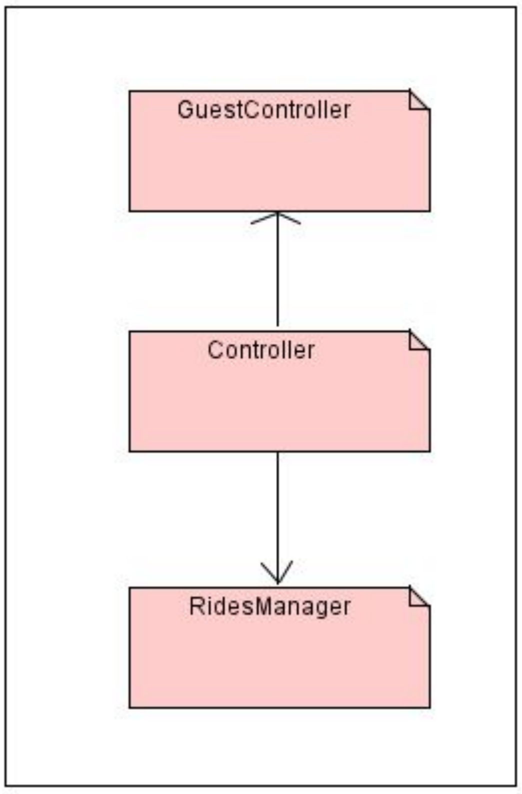


2.4.2 Subsystem Integration Sequence

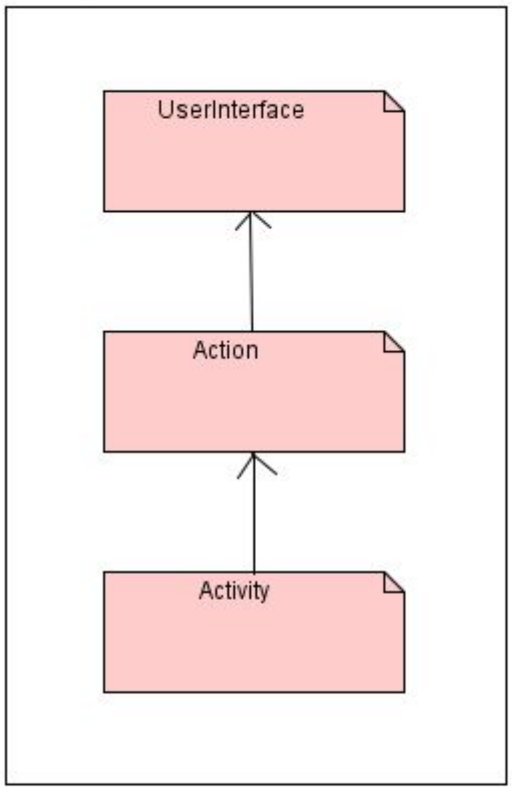
Components of Base Networking



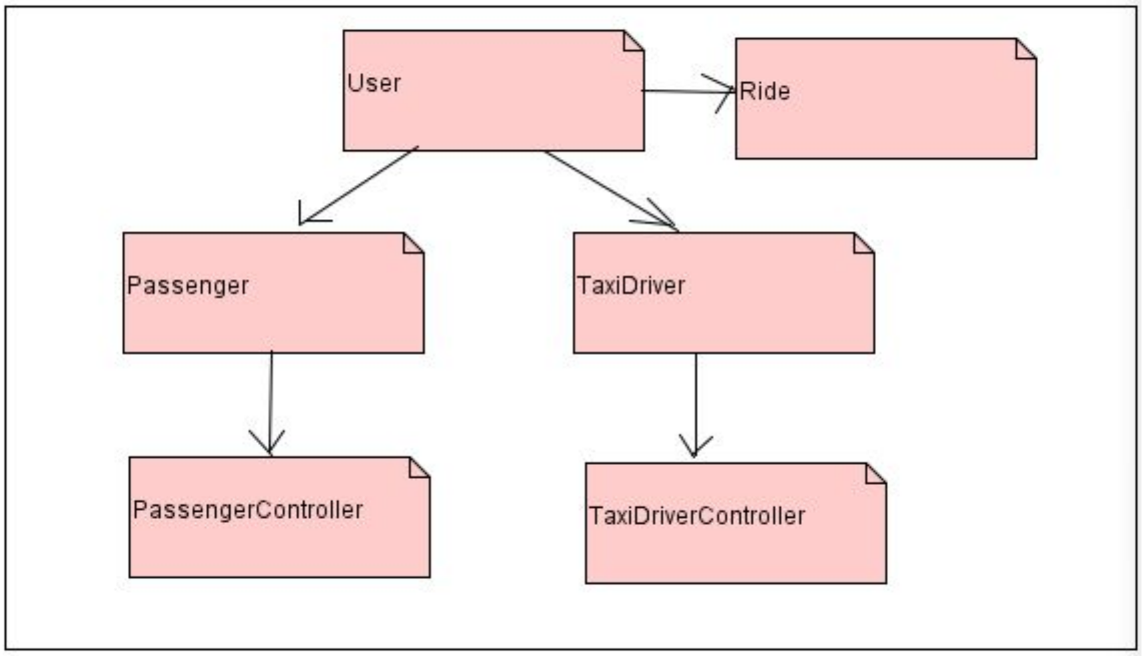
Components of Base Server

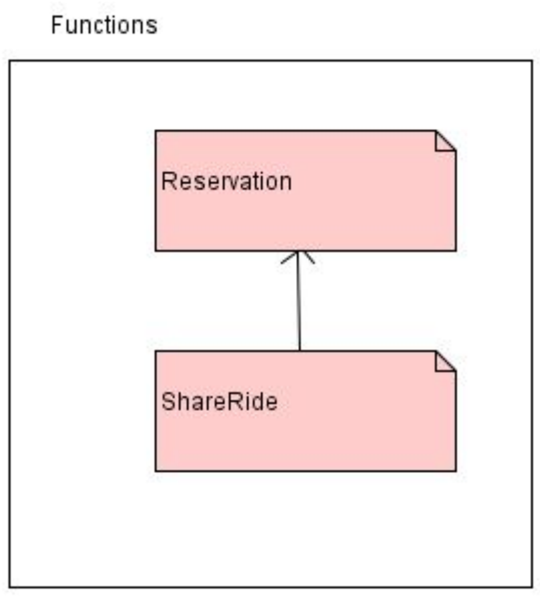


Components of Base Client



Extended User Components





3.Individual Steps and Test Description

Test Case Identifier	I-1
Test Items	Ridesmanager →Controller
Input specification	Create the typical Ridesmanager input
Output specification	Check if the correct methods are called in the Controller
Environmental needs	Rides manager driver

Test Case Identifier	I-2
Test Items	Controller →ClienteController
Input specification	Create the typical Controller input
Output specification	Check if the correct methods are called in the ClientController
Environmental needs	I-1 success needed

Test Case Identifier	I-3
Test Items	Activity →Action,Taxirequest Action
Input specification	Create an Activity
Output specification	Check that the Activity does create the correct Actions
Environmental needs	An Activity driver

Test Case Identifier	I-4
Test Items	Passenger driver → PassengerManager
Input specification	Input from Passenger
Output specification	Check if the right method is called in the

	PassengerManager
Environmental needs	An Passenger driver

Test Case Identifier	I-5
Test Items	ServerNetworkInterface →ServerMessage
Input specification	Invoke various types of network methods
Output specification	Check that the correctness of ServerMessage
Environmental needs	ServerNetworkInterface deriver

Test Case Identifier	I-6
Test Items	User→Ride
Input specification	Add a new Ride to an User
Output specification	Check that the Ride is correctly added
Environmental needs	User driver

Test Case Identifier	I-7
Test Items	Textdrver→TexidriversController
Input specification	Add a new Taxidriver to a TaxidriversController
Output specification	Check that the Taxidriver is correctly added
Environmental needs	TaxidriversController driver

4.Tools and Test Equipment Required

JUnit : For the implementation of unit tests it's an obvious choice, given its integration with the major IDEs and the overall simplicity and familiarity for the developers

Arquillian : For the integration testing phase, we have chosen this tool, since it easily integrates with Maven and JUnit.

5.Program and Test Equipment Required

In order to produce useful result during the integration test, these are stubs and data required that we generate automatically: We use a

random function that generates predefined response for driver availability. This function is used when, in the integration test, must be tested RequestAnswer module in DriverClientSubsystem.
We have generated a list with a lot of addresses of Milan metropolitan area, date and hour that are used to generate new Immediate or Booked Request. This list is used, particularly, when, during the integration test, ImmediateRequestForm and BookedRequestForm of the subsystem CustomerClient and RequestManagement and ZoneManagement of the subsystem System are tested.UserRegistry database is populated by users generated casually by a function (name, surname and other useful information used during the test).