Politecnico di Milano
A.A. 2015-2016
Software Engineering 2

Integration Test Plan

Alessandro Macchi
Caterina Finetti
Simone Manzoli

Abstract

This document describes the Integration Test Plan (ITP) for the Taxy App project.

This document contains the description of the integration tests for the project. This project is one of five assignments for the course SOFTWARE ENGINEERING 2 at Politecnico di Milano.

Document Title: Integration Test Plan

Version	Date	Author(s)	Summary
0.1	22/01/2016	A. Macchi, C. Finetti, S. Manzoli	Document creation

Sommario

1 Introduction

1.1 Purpose

This document describes the plans for testing the integration of the created components. The purpose of this document is to test the interfaces between the components as described in test report. Every team member who cooperates in the integration tests should read this document

1.2 Scope

The program allows users to access an app to make a reservation for a taxi and the taxi drivers to stay in the queue for the zone they belong and receive calls from users. The program's database contains user's and taxi driver's personal information, this information is used by the program to provide the service.

1.3 Definitions, acronyms and abbreviations

User: is a person who is registered in the database of the application. He has access to all the functions of the program that involves the requiring of a taxi, shared or not. He also has the possibility to save a list of preferred locations, that he can automatically choose when the System require from him an address as starting position or destination.

Guest: is a person who is using the application but is not registered in the database. He has access only to the registration functions.

User Information: all the information that concern a user; most of them have to be inserted during the registration (Name, Surname, tel. Number, e-mail, password), some of them can be inserted at any time after the registration (such as the personals locations) and others are assigned by the system (for example the number of blank-calls or the feedback).

Feedback: the feedback measures the reliability of a user. Is a simple relation between the total number of calls and the number of blank calls that a user have made (so a feedback equals to 1 means that he never missed a call).

Basic User Information: The information that a taxi driver visualizes when he receive a call. They are: Name, Surname, Feedback, Telephone Number of the user.

Blank Call: we define Blank-call a call for a taxi where the client is not at the starting location when the taxi arrives with a maximum late of X minutes, or a call that the user cancel before X minutes.

Missed Call: we define Missed-call a call for a taxi where the client is not at the starting location when the taxi arrives (X+1) or more minutes late.

Partner: someone who share a run with a user

Pick-up place: the Address where a user asks a taxi to come

Taxi Driver: a registered Taxi Driver

BCE diagram: boundary-controller-entity diagram

UX diagram: User experience diagram

1.4 Documents

RASD: Requirements analysis and specification

DD: Design Document

1.5 Overview

In the second chapter the items to be tested are mentioned. A specification for each test case is given in the third chapter. The fourth chapter specifies the procedures for these test cases.

In the fifth chapter the reports for all test cases are presented.

2. Integration Strategy

2.1 Entry Criteria

Before the integration test every function must be tested individually. The code needs to be complete and functional in its entirely. Every known exception needs to be treated and the developer must have a copy of the hardware that will be used (one driver's device witch gps and taximeter and one Smartphone).

2.2 Elements to be Integrated

The sub-system that will be integrated and tested are:

Call sub-system

Shared-call cost sub-system

User sub-system

Manage location sub-system

Manage account sub-system

Manage settings sub-system

Log in sub-system

Sign up sub-system

Every sub-system is analyzed in the DD.

2.3 Integration Testing Strategy

The items to be tested consist of the integration of the code modules developed, for the MyTaxiDrive project.

For testing we choose the bottom-up approach. This means that integration testing starts at the bottom level. This way the project will be built up from the bottom level.

2.4 Sequence of Component Integration

2.4.1 Software Integration Sequence

The first sub-system that will be tested is the Sign up and login sub-system in the test 1.1 and 1.2 is tested if the user created, or the access try will correspond to data entered by the user. Will be tested the client communicator, client translator, client manager and database manager.

The second sub-system will be the location manager sub-system, in the 2.1,2.2, and 2.3 test we will test tested the client communicator, client translator, client manager, database manager, agent communicator, agent translator, agent manager and I/O processor.

The third is the account manager sub-system the processes tested are equals that the previous.

The most difficult and important test are the 4.1.1, 4.1.2, 4.2, and 4.3 we will test the call sub-system and over the previous processes we will test the location process and the integration with google maps and the taximeter.

The last sub-system is the setting ones and we will test only the client translator and database manager.

3 Test case specifications

3.1 Integration test case 1.1

Test Object	Creation of a new User
Test number	1.1
Test Type	Manual test
Expected result	Creation of a new user on the database
Preconditions	No preconditions
Input	Access to the registration page Complete the fields with the requested informations: name, surname, telephone number, email, password Click on sign up button
Particular cases	This test should be repeated to test all the possible exceptions. Here follows the constraints: There must be no blank fields The birth date must be valid The e-mail must be valid The user must not be already registered

3.2 Integration test case 1.2

Test Object	User login
Test Number	1.2
Test Type	Manual test
Expected result	Login of an already registered User
Preconditions	Test 1.1 successfully executed, creation of the User executed
Input	Access to the login page Insert username and password in the fields Click on sign in button
Particular cases	This test should be repeated to test all the possible exceptions, listed below: The user does not exist on the database The password is wrong for the selected user

3.3 Integration test case 2.1

Test Object	Insert new location
Test Number	2.1
Test Type	Manual
Expected result	The user have a new personal location
Preconditions	-Test 1.2 successfully executed, a user can login to his page
Input	Access to location page Click on Add Location button Fill the blank fields with the address Confirm
Particular cases	This test should be repeated to test all the significant exceptions, listed below: The address is not valid A location with the same name already exists

3.4 Integration test case 2.2

Test Object	Modify location
Test Number	2.2
Test Type	Manual
Expected result	The location is modified correctly
Preconditions	-Test 1.2 successfully executed, a user can login to his page -Test 2.1 successfully executed, a user can create a location
Input	Access to location page Select an existing location Click on modify button Fill the fields with the new information Confirm
Particular cases	No particular cases

3.5 Integration test case 2.3

Test Object	Delete Location
Test Number	2.3
Test Type	Manual test
Expected result	The location is deleted successfully
Preconditions	-Test 1.2 successfully executed, a user can login to his page -Test 2.1 successfully executed, a user can create a location
Input	Access to location page Select an existing location Click on delete button
Particular cases	No particular cases

3.6 Integration test case 3.1

Test Object	Modification of user information
Test number	3.1
Test Type	Manual test
Expected result	User information modified
Preconditions	-Test 1.2 successfully executed, a user can login to his page
Input	Access to the account options page Click on Modify button Insert a new Tel. Number and a new email Confirm
Particular cases	This test should be repeated to test all the possible exceptions. Here follows the constraints: The email must be valid

3.7 Integration test case 4.1.1

Test Object	Immediate call
Test Number	4.1.1
Test Type	Mokito
Expected result	The process of making a call is completed successfully
Preconditions	-Test 1.2 successfully executed, a user can login to his page -Mokito should simulate the taxi driver reactions, accepting the call immediately and send an estimated arrival time of 5 minutes.
Input	Access to the calls page Select immediate call Choose an address and confirm Wait for acceptation Confirm the call when the arrival time is visualized on the screen
Particular cases	This test should be repeated to test all the significant exceptions, listed below: The address is not valid

3.8 Integration test case 4.1.2

Test Object	Immediate call taxi side
Test Number	4.1.2
Test Type	Mokito
Expected result	The process of accepting a call is completed successfully
Preconditions	-Mokito should simulate the user behavior, sending a call and accepting immediately the estimated arrival time. - Mokito should simulate the Queue behavior, deleting and adding when necessary the driver from it.
Input	Wait for the incoming call Click on Accept button Wait for the moked user reaction Finally close the call and free the taxi
Particular cases	This test should be repeated to test the call refusal.

3.9 Integration test case 4.2

Test Object	Delayed call
Test Number	4.2
Test Type	Mokito
Expected result	The process of making a call is completed successfully
Preconditions	-Test 1.2 successfully executed, a user can login to his page -Mokito should simulate the interaction with the calendar, making a fast jump to the selected date -Mokito should simulate the taxi driver reactions, accepting the call immediately and send an estimated arrival time of 5 minutes.
Input	Access to the calls page Select Delayed call Choose an address and confirm Wait for the notification on the date selected
Particular cases	This test should be repeated to test all the significant exceptions not tested befor, listed below: The data is not valid

3.10 Integration test case 4.3

Test Object	Shared call
Test Number	4.3
Test Type	Automatic
Expected result	The process of making a call is completed successfully
Preconditions	-Test 1.2 successfully executed, a user can login to his page -Test 4.1 and 4.2 successfully executed - Mokito should simulate the taxi driver reactions -Mokito should simulate the second user behavior, adding him to the list of users searching for a shared call for the same destination
Input	Access to the calls page Select Shared call selecting a waiting time of 10 minutes Insert the destination Wait for the confirmation, that should arrive when the system receive the request from the mocked user Confirm the call
Particular cases	This test should be repeated to test all the significant exceptions not tested before, listed below: The destination address of the users is not in the same zone The insert address is not valid The user does not confirm the call at the end

3.11 Integration test case 5.1

Test Object	Past calls data modification
Test number	5.1
Test Type	Manual test
Expected result	Past calls are correctly displayed
Preconditions	-Test 1.2 successfully executed, a user can login to his page - Test 4.1.1 successfully executed - At least 2 calls for a user created
Input	Access to the options page Click on Past calls date Modify the date Confirm
Particular cases	This test should be repeated to test all the possible exceptions. Here follows the constraints: The date must be valid

3.12 Integration test case 5.2

Test Object	Language modification
Test number	5.2
Test Type	Manual test
Expected result	The new language is correctly displayed on all pages
Preconditions	-Test 1.2 successfully executed, a user can login to his page
Input	Click on Language Choose a language Confirm
Particular cases	No particular cases

4 Test procedures

4.1 Manual test

In the following descriptions of our manual test if some parts are omitted is to avoid repetition. The prerequisites of tests are implied correct.

4.2 Automatic test

The test case with automatic tool have been created after the entair development of the myTaxyDrive application so we not follow development strategies like Test-driven development (TDD). We essentially use a Waterfall Model for software development process. The waterfall model is a sequential design process composed by the following step:

1. Requirements specification resulting in the product requirements document
2. Design resulting in the software architecture
3. Construction (implementation or coding) resulting in the actual software
4. Integration
5. Testing and debugging
6. Installation
7. Maintenance

The structure of the software not allow to create a complete test case suite but we try to test the most critical function. For the other test case not covered by the automatic tests whe use a manually stategies as described in the third part of the document.

For the realization of authomatic test we use essentialy this tools:

Junit: The unit testing framework for the Java programming language

Mokito: An open source testing framework that allows the create mock object. Mock objects are simulated objects that mimic the behavior of real objects in controlled ways. A programmer typically creates a mock object to test the behavior of some other object.

Arquillian: A test framework that can be used to perform testing inside a remote or embedded container, or deploy an archive to a container so the test can interact as a remote client.