POLITECNICO DI MILANO

Software engineering 2 project

TAXI SERVICE

INTEGRATION TEST PLAN DOCUMENT

AUTHOR

Pallotta Simone

PROFESSOR

Raffaela Mirandola

Version 1.0

ACADEMIC YEAR 2015/2016

intentionally left blank

CONTENTS

Introduction 4

1.1 Revision History 4

1.2 Purpose and Scope 5

1.3 List of Definitions and Abbreviations 6

1.3.1 Definitions 6

1.3.2 Acronyms 6

1.4 List of Reference Documents 7

Integration Strategy 8

2.1 Entry Criteria 8

2.2 Elements to be Integrated 9

2.3 Integration Testing Strategy 9

2.4 Sequence of Component/Function Integration 10

2.4.1 Software Integration Sequence 10

Individual Steps and Test Description 13

Tools and Test Equipment Required 15

Introduction

1.1 Revision History

Document Title	Integration Test Plan
Authors	Simone Pallotta
Version	1.0
Document Status	Under revision

1.2 Purpose and Scope

This document shows the plans for testing the integration of the single components (Client, server) and the compound system composed by both.

The purpose of this document is threefold:

- testing the web services

- testing the client

- perform tests run on mobile that imply response from server, results will be verified on client

The web services will be testing using *two phase* approach:

1. For every services we will write *shell curl command* that adopt REST protocol to trigger request on the server. This curl commands will be executed in batch mode inside shell script. This script will compare the result with known values. All this single script will be run sequentially in the server test suite script
2. Equivalent curl commands will be issued on the mobile platform using native API implementing RESTFUL calls in the way that must be the same as performed via single curl scripts.

1.3 List of Definitions and Abbreviations

1.3.1 Definitions

curl: C command line tool to invoke remote url

REST: Representational state transfer, standard approach to simply HTTP request

Push Notification: a short service message sent to mobile Apps with no assurance of delivery. Such message can be received even if the specific app is not running. The system take charge to save and deliver it to user / to the app later.

Push Notification Service: system to deliver mass Push Notifications to mobile Apps

SMS Gateway: external system in charge of sending confirmation code via telephone company

1.3.2 Acronyms

A**PNS:** Apple push notification system/servers

DB: database

1.4 List of Reference Documents

- Requirement Analysis and Specification Document (RASD)
- Design Document (DD)
- web document from https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/UnitTesting.html for iOS version
- web document from http://developer.android.com/tools/testing/index.html for android version

Integration Strategy

2.1 Entry Criteria

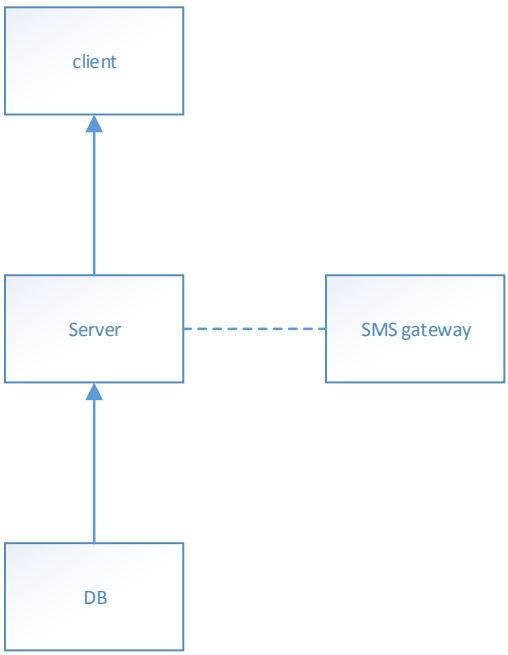
Before testing web services we must sure that a valid connection can be established to the database.

This can be accomplished by using script containing a *Telnet* command on port 3306.

Once DB is up, we can test all sub-system that runs on server.

Since our systems architecture is made up of four tier as described in DD we have introduced an order of integration test as showed at figure below:

Lower tier must be tested before proceed on upper level.



2.2 Elements to be Integrated

we must integrated five different pieces:

- DB

- web services

- SMS gateway

- APNS

- mobile client

Refer to our DD for an in-depth description of overall architecture and mutual interaction between the above parts

A big challenge is represented by testability/integration of external parts as they are not under our control, so we can integrated them but cannot assure the correctness their behaviour: we make the assumption that their programmers have conducted perform such test on their own.

2.3 Integration Testing Strategy

To test integration we decided to use the bottom-up approach, this means that the lowest level components are tested first.

After the integration testing of lower levels, the next level will be tested for integration.

One important point about integration concerns the order of testing between server components and client components.

In this approach server components must be considered the bottom level in respect of client components as webs services can be considered parts client is based on.

We decided to use bottom-up approach since it resemble the big processor of application: starting building DB and web services and after we developed the client.

An other key point is the integration with external modules: as specified in DD our system is connected to different external infrastructures SMS gateway and APNS network.

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

We identified the following modules:

- mobile client

- web server

The following modules will be integrated but not tested since are external components

- SMS gateway

- APNS

Components of the mobile client:

Connection manager
URL request manager
keychain manager / user setting manager
push notification manager
geo location manager

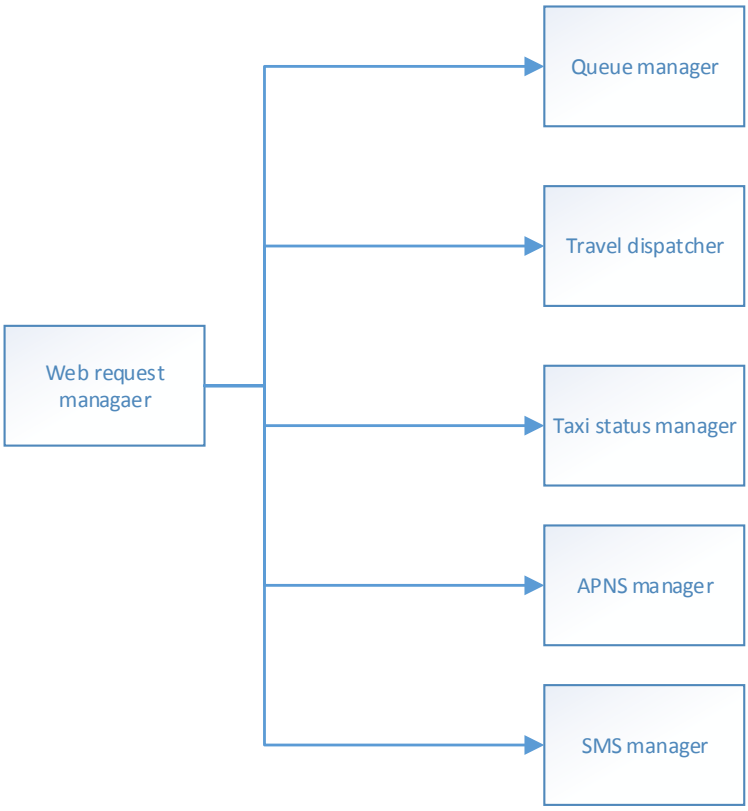
Components of the web server:

Web request manager
Travel dispatcher
queue manager
SMS manager
APNS manager
Taxi status manager

2.4.2 Subsystem Integration Sequence

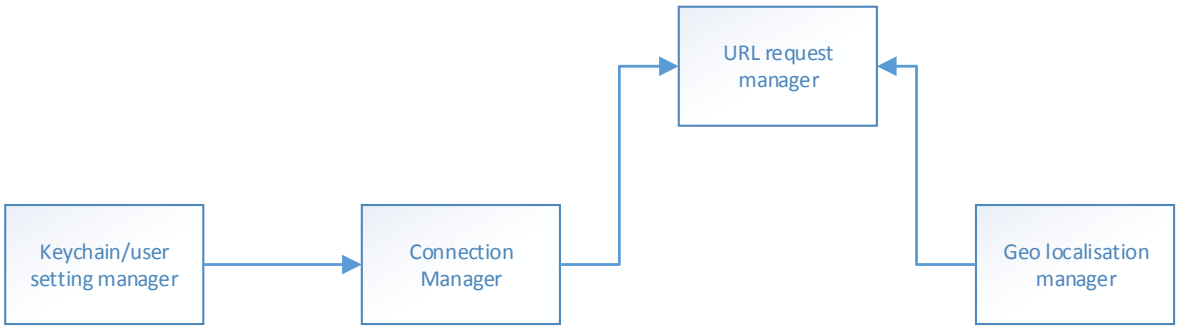
Integration test of the web server:

I1	Web request manager —> Travel dispatcher
I2	Web request manager —> queue manager
I3	Web request manager —> taxi status manager
I4	Web request manager —> SMS manager
I5	Web request manager —> APNS manager



Integration test of the mobile client:

I6	keychain manager/user setting manager —> Connection Manager
I7	Connection manager —> URL request manager
I8	geo location manager —> URL request manager



Individual Steps and Test Description

Test Item(s)	Web request manager —> Travel dispatcher
Input Specification	create JSON post request
Output Specification	Check if the travel dispatched is generated

Test Item(s)	Web request manager —> queue manager
Input Specification	create JSON post request
Output Specification	Check if the taxi is managed in queue correctly

Test Item(s)	Web request manager —> taxi status manager
Input Specification	create JSON post request
Output Specification	Check if the taxi status is updated correctly

Test Item(s)	Web request manager —> SMS manager
Input Specification	create JSON post request
Output Specification	check if SMS gateway answer OK

Test Item(s)	Web request manager —> APNS manager
Input Specification	create JSON post request
Output Specification	check if feedback manager contains message id

Test Item(s)	keychain manager/user setting manager —> Connection Manager
Input Specification	create test user and password
Output Specification	check if connection answer correctly

Test Item(s)	Connection manager —> URL request manager
Input Specification	create JSON request
Output Specification	check if request is parsed and converted in query

Test Item(s)	geo location manager —> URL request manager
Input Specification	(input from HW)
Output Specification	check if geo loc data is present

Tools and Test Equipment Required

Command line script on server side for testing request and response of REST web services.
For android version we used JUnit: most popular framework for java programming used for writing both unit and integration tests.

For iOS version we used Continuous Integration present in Xcode : https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/xcode_guide-continuous_integration