Taxi Service
Acceptance Test Plan

Version 1.1

Revision History

Date	Version	Description	Author
2012-12-28	0.1	Initial Draft	Luca Zangari
2012-12-29	0.2	Chapter 1, 10 Draft	Igor Piljić
2012-12-29	0.3	Chapter 11, 12, 13 Draft	Igor Piljić
2012-12-29	0.4	Chapter 2,8,9 Draft	Jelena Jerat
2012-12-29	0.5	Chapter 1, 4 Draft	Luca Zangari
2012-12-30	0.51	Chapter 2 edited	Leon Dragić
2012-12-30	0.52	Chapter 3 edited	Leon Dragić
2012-12-30	0.6	Chapter 4, 5, 6 edited	Leon Dragić
2012-12-30	0.61	Chapter 6.1 edited	Leon Dragić
2012-12-30	0.62	Chapter 2,3 edited	Jelena Jerat
2012-12-30	0.7	Chapter 7 Draft	Marko Coha
2012-12-30	0.71	Chapter 10 edited	Karlo Zanki
2012-12-31	0.72	Chapter 10.1 edited	Jelena Jerat
2012-12-31	0.8	Chapter 4, 7.1 edited, General revision	Luca Zangari
2012-12-31	0.9	Chapter 10.2	Jelena Jerat
2012-12-31	1.0	General revision	Leon Dragić
2013-01-20	1.1	Polishing up language, filling in gaps	Lyudmil Angelov

Table of Contents

1. Introduction 4
   - 1.1 Purpose of this document 4
   - 1.2 Intended Audience 4
   - 1.3 Scope 4
   - 1.4 Definitions and acronyms 4
     - 1.4.1 Definitions 4
     - 1.4.2 Acronyms and abbreviations 4
   - 1.5 References 4

2. Test-plan introduction 5

3. Test items 5
   - 3.1 Catch a Cab (Customer client Android Application) 5
   - 3.2 Taxi Client Android Application 5
   - 3.3 Server 5

4. Features to be tested 5

5. Features not to be tested 6

6. Approach 6
   - 6.1 Approach to configuration and installation 6

7. Item pass/fail criteria 7
   - 7.1 Installation and Configuration 7
   - 7.2 Documentation problems 7

8. Suspension criteria and resumption requirements 7

9. Environmental needs 8
   - 9.1 Hardware 8
   - 9.2 Software 8
   - 9.3 Other 8

10. Test procedure 8
    - 10.1 Test case specifications 8
      - 10.1.1 Customer Client Test Case – CCT 8
      - 10.1.2 Taxi Client Test Case – TCT 10
      - 10.1.3 Server Test Case – ST 11
    - 10.2 Test plan 14

11. Responsibilities 14
    - 11.1 Developers 14
    - 11.2 User representative 14

12. Risks and contingencies 15

13. Approvals 15

1. Introduction

1.1 Purpose of this document

The objective of acceptance testing is to confirm that the developed application meets its requirements and to ensure that the system works correctly and is usable before it is formally delivered to the end user. This document is the acceptance test plan for the three main components of the Taxi Service system: the Taxi Client, the Central Server, and the Customer Client. It describes the scope of the work performed and the approach taken to execute the tests created to validate that the system performs as required.

1.2 Intended Audience

This document is intended for the:

- Project customer and supervisor,
- Team members,
- All persons who are responsible for monitoring the project.

1.3 Scope

This document covers all tests cases for Taxi service android applications and server. It also includes results of those tests, as well as techniques and tools needed for testing.
It doesn't include test of network or external systems that Taxi Service uses (such as Google maps API, 3G network connection).

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
Taxi service	Project name
Android	Mobile operating system
Catch A Cab	Android application for the customers
Dispatch	Android application for the taxi drivers

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
GPS	Global positioning system
AVD	Android Virtual Device

1.5 References

- TaxiService Website
- TaxiService Project Vision
- TaxiService Requirement Definition
- TaxiService Design Description
- TaxiService Project Plan

2. Test-plan introduction

The consumer-facing Catch A Cab Android application, the taxi-driver-facing Dispatch Android application, and the main server component will be tested before the final delivery. Testing will be conducted in several levels. Every server and Android application component will be unit tested using unit testing tools integrated in IDE-s (low level testing). Server will also be stress-tested using Apache JMeter tool which can be used for simulation of a heavy load on the server. Finally, user acceptance testing will ensure that end users can use Taxi service applications the way they were described in the use-cases. It is desirable that each component is tested by a third party or at least a team member that wasn't involved in the component development. Both client applications will also be tested from a third party in order to test the user interface to verify is it user friendly. The main goal of testing the applications is to verify that application meets the requirements described in Requirements Definition document.

3. Test items

In the user acceptance testing phase we have identified two main test items: the Taxi Client application and Customer Client application. Most of the application logic contained in the central server will be tested indirectly by testing the other two components, which use the server as a means of exchanging messages. Although most of the server functionality can be tested using the two clients, there are some specific cases that have to be tested separately (testing the server directly). In addition to this, unit testing will take place on every of these components.

3.1 Catch a Cab (Customer client Android Application)

Catch a Cab is Android application intended for Taxi Service customers who want to order a taxi to a specified location. The application communicates with the server application using RESTful API. During the acceptance testing Catch a Cab application will be tested to verify that it meets all functional and non-functional requirements defined in the Requirements Definition document.

3.2 Taxi Client Android Application

Taxi Client Android application is intended for taxi drivers who want to pick up the customers using Taxi Service. The application communicates with the server application using RESTful API the same as the Catch a Cab application does. It will also be tested to verify that it meets functional and non-functional requirements.

3.3 Server

Taxi Service server application is a RESTful web service which is used by both client Android applications to provide their main functionality. Server will be tested to verify that it meets both functional and non-functional requirements. Most of the server functional requirements will be tested through two client applications. One of the most important non functional requirements, ability of the server to serve large number of clients (both taxi and customer clients) at the same time, will be tested using stress testing tool JMeter.

4. Features to be tested

Features that will be tested are as following:

Identity	Sta Tus	Prio Rity	Description
			Mobile application for taxi user shall be able to:
TCT-01 TCT-02 TCT-03	Done	**M**	Communicate with the server side, sharing repeatedly current position and status
TCT-04 TCT-05	Done	**M**	Receive from the server side the order requests, accept or decline it
TCT-04	Done	**M**	Receive information about the orders

			Server shall be able to:
	To Do	**S**	Work under high-load stress
ST-01 ST-02	Done	**M**	Receive constantly the taxi positions and statuses
ST-06	Done	**M**	Divide the city in different areas and make a queue for every area
ST-06 ST-07	Done	**M**	Maintain the queues with the taxi positions and statuses
ST-03 ST-04	Done	**M**	Receive orders from customer clients, and dispatch them to the nearest queues
			Mobile application for customer user shall be able to:
CCT-01 CCT-02 CCT-04	Done	**M**	Send an order for a taxi
CCT-05 CCT-06 CCT-07 CCT-08	Done	**S**	Receive information about the taxi which will pick the customer up and its time of arrival
CCT-03	Done	**S**	Automatically get the phone number from the SIM card or allow the customer to enter it manually

Requirement priority:

M – Must have this requirement to meet the business needs.

S – Should have this requirement if possible, but project success does not rely on it.

C – Could have this requirement if it does not affect anything else in the project.

5. Features not to be tested

The user acceptance testing phase will not test most of the server functionality directly. Instead, the server application will be tested indirectly – as a result of users' interaction with the mobile applications.

The system will not be tested for malfunctions on the physical level such as power outages, high network delays, etc.

6. Approach

The SCRUM model includes a testing phase in every sprint. Throughout the development process, features were tested before and after their integration in the system. The QA team of 2 people was also established and their task was to test the whole system additionally in order to find bugs which went unnoticed to the developers of that system part. The QA team changes its members for every sprint. Before the final delivery, additional testing such as stress testing will take place in order to deliver fully working product.
Many tools were used for testing the system – Robotium, JUnit, MS VS integrated unit testing, JMeter, Fiddler, etc.

6.1 Approach to configuration and installation

The main server is deployed with MS VS 2012 on Microsoft Azure which saves much time which is usually spent on configuration and installation. The system is also deployed locally. Android clients are delivered as

.apk files and are easily installable to Android devices or AVD Emulator.

7. Item pass/fail criteria

The general testing criteria:

- Whenever possible, acceptance tests will be automated using the Robotium framework. If a test is automated, it passes if it compiles, runs, and passes all of its assertions. It is considered a failing test otherwise.
- If a test case is not automated and is done properly, i.e. the preconditions are met and all input data is as defined by the input definitions, the test is considered a pass if the result is according to the output definition.
- If a test case is not automated and done properly, but the result isn't as described in the output definition, the test is considered a fail.
- If the test isn't done properly, it cannot be said if the test has passed or failed.

7.1 Installation and Configuration

The server side of the application must be properly installed and set up for testing. In order to test each of the clients, an Android application must be installed on a compatible device. The requirements for the device are a working GPS module and an internet connection.

7.2 Documentation problems

Documents that can affect testing are the Requirements specification document, because it defines all the necessary requirements which have to be fulfilled, and the Design description document, because it defines software design in detail. However, those documents were reviewed in the previous project iterations, so there is currently no need to do any major changes which would affect testing. Other documents can't affect testing in any way.

8. Suspension criteria and resumption requirements

During the testing period people who are performing the testing should be in touch with developers of the applications or server component. Testing can be suppressed and then continued from the same point in following situations:

- A bug on the client applications that doesn't affect the server
- A bug in the server functionality that doesn't affect more than one use-case
    - In this particular case testers should be very careful, because if the bug affects more than one use case it could also affect other tests that were previously run and therefore testing process has to be started from the beginning
- If the server goes down for some reason

Before continuing with the testing the following requirements should be fulfilled:

- The cause of test suspension is no longer valid if:
    - Bugs are fixed
    - Server is up again and nothing was changed in server side logic

If the bug is found in the test scenarios which involve more than one system component it is very risky to continue testing from the same point after fixing the bug. Almost every major change of the server side logic should mean that testing should be restarted from the beginning.

9. Environmental needs

9.1 Hardware

- Two Smartphones or tablets with Android OS and GPS chip
- Server computer(s)

9.2 Software

- Android OS on the Smartphone
- Windows server or MS Azure account
- MVC4 framework
- Microsoft Entity framework
- MS SQL Database Server
- MS Visual Studio 2012
- Eclipse IDE
- Android Emulator

9.3 Other

- Internet connection
- Server should be running the Taxi service server application

10. Test procedure

10.1 Test case specifications

10.1.1 Customer Client Test Case – CCT

10.1.1.1 Making order – CCT-01

Description	User makes an order at his current location
Test type	Positive
Precondition	Device connected to internet and has GPS turned on. Customer defined his phone number. Customer orders taxi in Milano.
Input definition	• User opens “Catch a cab” application • User clicks “Call a cab” button • User clicks “Send order” button • User clicks “Call” button on popup window
Output definition	User gets message “Your order has been received”. Server gets new order.
Remarks	

10.1.1.2 Making order – CCT-02

Description	Customer makes an order without specified location
Test type	Negative
Precondition	Device not connected to internet or GPS is turned off.
Input definition	• User opens “Catch a cab” application • User clicks “Call a cab” button • User clicks “Send order” button • User clicks “Call” button on popup window
Output definition	An error message “No GPS available, please try again” is displayed to the user. Server doesn’t get new order
Remarks	

10.1.1.3 Making order – CCT-03

Description	Customer makes an order with no phone number
Test type	Negative
Precondition	Device connected to internet and has GPS turned on. Device doesn't have SIM card or user didn't input his phone number in settings
Input definition	• User opens "Catch a cab" application • User clicks "Call a cab" button • User clicks "Send order" button • User clicks "Call" button on popup window
Output definition	An error message "No phone number provided, please put a SIM card in your device or enter the number manually" is displayed to the user. Server doesn't get new order
Remarks	

10.1.1.4 Making order – CCT-04

Description	Customer makes an order outside of Milano
Test type	Negative
Precondition	Device connected to internet and has GPS turned on. Customer defined his phone number. Customer orders a taxi that is out of Milano.
Input definition	• User opens "Catch a cab" application • User clicks "Call a cab" button • User clicks "Send order" button User clicks "Call" button on popup window
Output definition	An error message "Out of Milan" is displayed to the user. Server doesn't get new order
Remarks	

10.1.1.5 Checking current order – CCT-05

Description	Customer checks position of taxi that is assigned to pick him
Test type	Positive
Precondition	Device connected to internet. Customer already has pending order
Input definition	• User opens "Catch a cab" application • User clicks "Messages" button
Output definition	Map with marker showing taxi and customer location is shown to customer
Remarks	

10.1.1.6 Checking current order – CCT-06

Description	Customer checks position of taxi that is assigned to pick him
Test type	Negative
Precondition	Device not connected to internet.

Input definition	• User opens “Catch a cab” application • User clicks “Messages” button
Output definition	An error showing “No response from server” is shown to customer
Remarks	

10.1.1.7 Checking current order – CCT-07

Description	Customer checks position without ordering taxi
Test type	Negative
Precondition	Device connected to internet. Customer doesn’t have pending order.
Input definition	• User opens “Catch a cab” application • User clicks “Messages” button
Output definition	An error showing “Customer doesn’t have any pending order” is shown to customer
Remarks	

10.1.1.8 Checking current order – CCT-08

Description	Customer checks detailed information about taxi that is assigned to pick him
Test type	Positive
Precondition	Device connected to internet. Customer has pending order.
Input definition	• User opens “Catch a cab” application • User clicks “Messages” button • User clicks on a marker representing taxi on a map
Output definition	New window pops up in which taxi name and time of receiving order are shown
Remarks	

10.1.2 Taxi Client Test Case – TCT

10.1.2.1 Changing status – TCT-01

Description	User changes current taxi status from “on duty“ to “off duty“
Test type	Positive
Precondition	Device connected to internet and the current status is “on duty”
Input definition	• User opens “Taxi client“ application • User clicks “on duty” button
Output definition	Taxi status changes to “off duty“.
Remarks	

10.1.2.2 Changing status – TCT-02

Description	User changes current taxi status from “off duty“ to “on duty“
Test type	Positive

Precondition	Device connected to internet and the current status is “off duty”
Input definition	• User opens “Taxi client“ application • User clicks “off duty” button
Output definition	Taxi status changes to “on duty“.
Remarks	

10.1.2.3 Inactive taxi goes to off duty – TCT -03

Description	Taxi status changes to “off duty” if taxi didn’t update his location for more than 5 minutes
Test type	Positive
Precondition	Device connected to internet and the current status is “free” or “busy”
Input definition	• User opens “Taxi client“ application • Taxi client application continuously sends current taxi location to the server • User disconnects Android device from the internet and stays disconnected for more than 5 minutes
Output definition	Taxi status changes to “off duty“.
Remarks	

10.1.2.4 Accepting order – TCT-04

Description	User accepts an order offered to him
Test type	Positive
Precondition	Device connected to internet and has GPS turned on. Taxi client application is up and running. Order is offered to taxi driver.
Input definition	• Popup window opens on “Taxi client“ application • User clicks “Accept“ button
Output definition	User gets message with order details. Taxi status is changed to busy.
Remarks	

10.1.2.5 Rejecting order – TCT-05

Description	User rejects an order offered to him
Test type	Positive
Precondition	Device connected to internet and has GPS turned on. Taxi client application is up and running. Order is offered to taxi driver.
Input definition	• Popup window opens on “Taxi client“ application • User clicks “Reject“ button
Output definition	Taxi is moved to the last position in the queue.
Remarks	

10.1.3 Server Test Case – ST

10.1.3.1 Detect Zone – ST-01

Description	Server can detect zone from given GPS coordinates

Test type	Positive
Precondition	GPS coordinates received.
Input definition	• Server gets the zone for received coordinates
Output definition	
Remarks	

10.1.3.2 Coordinates validation – ST-02

Description	Server detects invalid GPS coordinates
Test type	Negative
Precondition	GPS coordinates received.
Input definition	• Server detects invalid coordinates
Output definition	Error message is sent in server response
Remarks	

10.1.3.3 Make order – ST-03

Description	Server receives order request and creates new order in the system.
Test type	Positive
Precondition	Order request received.
Input definition	• Server gets the order request message • Server validates received request. • Server creates new order in the system
Output definition	Server responds with positive status message “ok”.
Remarks	

10.1.3.4 Assign order to first taxi in queue – ST-04

Description	Server receives order request and assigns it to first taxi in queue
Test type	Positive
Precondition	Order request received.
Input definition	• Server gets the order request message • Server validates received request. • Server detects zone from which the request was made • Server gets the first taxi from the queue for the specified zone • Server assigns order to that taxi • Taxi's status is changed to busy, and taxi is removed from the queue
Output definition	
Remarks	

10.1.3.5 Taxi with assigned order changes status to “off duty” – ST-05

Description	Taxi with assigned order changes status to “off duty”, order is offered to next taxi in queue

Test type	Positive
Precondition	Order assigned to the taxi. Taxi has accepted the order. Change status message received
Input definition	• Server gets the change status request message • Server changes taxi status to “off duty”, and taxi is removed from the queue • Server detects zone from which the request was made • Server gets the next taxi in the queue for the specified zone • Server assigns order to that taxi • Taxi's status is changed to busy, and taxi is removed from the queue
Output definition	
Remarks	

10.1.3.6 Taxi with assigned order changes status to “off duty” after order is offered to him, but before he accepted or rejected the order – ST-06

Description	Taxi with assigned order changes status to “off duty”, order is offered to next taxi in queue
Test type	Positive
Precondition	Order assigned to the taxi. Taxi hasn't accepted or rejected the order. Change status message received
Input definition	• Server gets the change status request message • Server changes taxi status to “off duty”, and taxi is removed from the queue • Server detects zone from which the request was made • Server gets the next taxi in the queue for the specified zone • Server assigns order to that taxi • Taxi's status is changed to busy, and taxi is removed from the queue
Output definition	
Remarks	

10.1.3.7 All taxis in queue go to “off duty” after order request is made – ST-07

Description	All taxis in queue go to “off duty”, server sends error message to client
Test type	Negative
Precondition	Order request is made.
Input definition	• Server gets the change status request message • Server changes taxi status to “off duty”, and taxi is removed from the queue • Server detects zone from which the request was made • Server gets the next taxi in the queue for the specified zone • Server assigns order to that taxi • Taxi's status is changed to busy, and taxi is removed from the queue • Server gets the change status request message from second taxi • Server changes taxi status to “off duty”, and taxi is removed from the queue

	• Server detects zone from which the request was made • Server detects that there are no more taxis in the queue for the specified zone
Output definition	Server responds with negative status message “No taxis in queue”.
Remarks	

10.1.3.8 Administrator can add taxi – ST-08

Description	Administrator can add new taxis to system
Test type	Positive
Precondition	Administrator is successfully logged in the web application
Input definition	• Administrator selects option “manage taxis” • Administrator fills in necessary data • Administrator presses button “Add” • New taxi is added to the system
Output definition	New taxi is visible under existing taxis.
Remarks	

10.1.3.9 Administrator can remove taxi – ST-09

Description	Administrator can remove taxi from system
Test type	Positive
Precondition	Administrator is successfully logged in the web application
Input definition	• Administrator selects option “manage taxis” • Administrator presses button “Delete” next to the taxi he wants to remove • Taxi is removed from the system
Output definition	Taxi is no longer visible under existing taxis.
Remarks	

10.2 Test plan

Server tests can be executed apart from the other tests, because they don’t depend on the test results of other parts of the system. In order for both applications to work properly it has to be confirmed that server application meets the requirements.
Some test scenarios are dependent on the others, so testing both applications will have to be done at the same time. For example, in order to execute test scenario TCT-04 (Accepting order), an order already has to be received, and therefore test scenario CCT-04 (Making order) has to be executed.

11. Responsibilities

11.1 Developers

- Unit testing every feature before integrating them into the system
- Testing whole system
- Fixing bugs

11.2 User representative

- Changing and adding requirements on time

- Report if any bugs are found

12. Risks and contingencies

- Android applications are tested for some smart phones and tablets, not for all, so there is possibility that applications won't work on some of them
  How to avoid risk: Check application on as many smart phones and tablets it is possible, work with widely used API's
- Not enough time to test the system and fix bugs
  How to avoid risk: Test every feature as soon as it is done; define QA team that will be responsible on checking eventual bugs; set strict deadlines on finishing and testing each feature
- Possibility of crashing the system when large number of customers access the server
  How to avoid risk: Simulate situation where large number of customers uses the system and check if everything is working as it should
- Not all aspects of the application are easily tested through automatic tools. For example, it may not be possible to mock a GPS location, so any feature that relies on getting a GPS fix may not be testable by automatic means
  How to avoid risk: Refactor and re-architect the design in order to enable easy mocking of components where possible in order to automate testing. Failing that, fall back on manual testing.

13. Approvals

Name	Title	Date yyyy-mm-dd	**Signature**
Elisabetta Di Nitto	Supervisor		