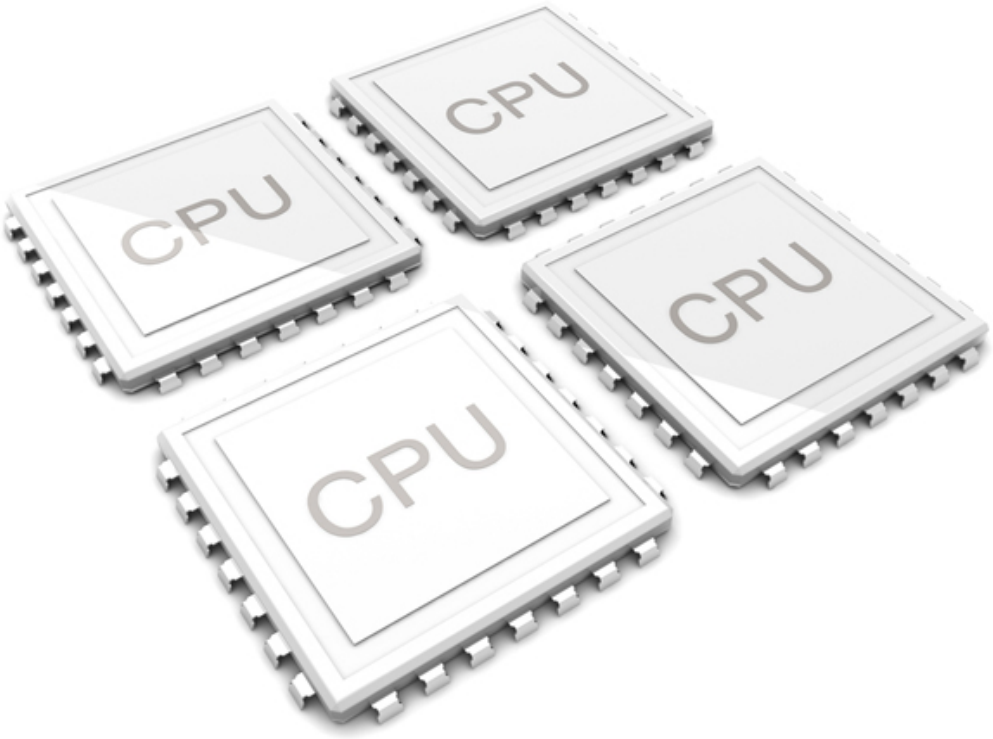
Testing Document

Project: Traffic Camera Image Analysis

Client: DPSS, CSIR

Team: Quadcore Productions

Author(s):	**Student number(s):**
Mpho Baloyi	14133670
Hlengekile Jita	14077893
Mayimela Moses	14019702
Mbhele Themba	14007950



University of Pretoria, Department of Computer Science

22 October 2016

1 Introduction

This document contains all the details of the tests that were conducted on the system. In addition to tests that have already been conducted, plans for futures tests will also be stated.

2 Purpose

The purpose of this document is to combine the unit tests of the system and the results of the unit tests (report) into a single document.

3 Plan for Testing (Android)

3.1 Scope

The features of the system that will be tested are the following:

- Route construction
- Communication with server (making a request)
- Communication with server (receiving a response)

3.2 Testing Environment

As the functions that are being used by the system were developed, tests were designed to test their functionality to verify that the functions do what they are intended to do so that as we continue with the development, we use functions that have been thoroughly tested and have been verified as correct. On the android side of the system, JUnit was used to conduct all unit tests. We made use of a testing framework because it allows us to automate the testing process as we do not have to manually run tests for each function that we intend to test. Instead, JUnit will run all of the tests for us provided that we have properly annotated our functions and made use of assertions.

3.3 Running the unit tests

To run the unit tests, an internet connection has to be established, in order to make server responses, and an IDE such as eclipse, android studio or IntelliJ must be used. Once that infrastructure is in place, and the tests have all been written, executing the tests simply involves right clicking on the folder

in which the unit tests are located and choosing the "run tests" options. this will execute all the tests in that folder and generate a report displaying the results of the tests. The tests will run across various operating systems.

3.4 Test Cases

3.4.1 Test Case 1: Route Construction (valid data)

The purpose of this unit test is determine whether the route construction function acts according to its specification when provided with data that it is possible to construct a route from.
Input:

- A start and end locations that are user specified

Outcome: For this test to be successful, the following result is expected

- A JSON object (Received from the Google server response)
- The JSON object must not be empty. It should include all the data needed to process and construct a route. Such data includes latitude/longitude values for various points on the route.

3.4.2 Test Case 2: Route Construction (invalid data)

The purpose of this unit test is to test whether the route construction function will deal with a scenario where valid input is used, but no routes (via vehicle) can be constructed due to nature obstructions such as oceans, inadequate map data, or garbage input from the user.
Input:

- A start and end locations that are user specified

Outcome: For this test to be successful, the following result is expected

- A JSON object (Received from the Google server response)
- The JSON object must be empty.

3.4.3 Test Case 3: Server Response

The purpose of this test is to determine test that our server will always generate a response that is in the expected format. The expect format is XML.
Input:

- Encoded poly line that includes all the latitude and longitude values of a route

Outcome: For this test to be successful, the following response is expected from the server

- An XML string

3.4.4 Test Case 4: Input validation

The purpose of this test is to make sure that the input data received from the user is processed and validated correctly
Input:

- Empty input data
- Valid input data (i.e Non-empty input data)

Outcome: For this test to be successful, the following results are expected

- If empty input data is provided by user, the function should return false so that the appropriate action may be taken.
- If the user provides non-empty data, the function should return true, signalling that valid data was provided. Other than checking for non-empty data, it would be difficult to determine whether the user has provided useful input because we cannot determine wheter a place exists or not. Instead, this will be taken care of in test case 2 when a request is made to the Google server.

3.5 Unit Test Report

The following results were obtained from running all the tests

3.5.1 Test Case 1

Test description: Route construction test with valid data
Result: Success.
Reason: A non-empty JSON object was returned.

3.5.2 Test Case 2

Test description: Route construction test with invalid data
Result: Success.
Reason: An empty JSON object was returned.

3.5.3 Test Case 3

Test description: Server response test
Result: Success.
Reason: An XML file was returned by the server.

3.5.4 Test Case 4.1

Test description: Input validation test with empty user input fields
Result: Success.
Reason: The validation function returned false.

3.5.5 Test Case 4.2

Test description: Input validation test with a valid start location and an empty end location
Result: Success.
Reason: The validation function returned false.

3.5.6 Test Case 4.3

Test description: Input validation test with empty start location and a valid end location
Result: Success.
Reason: The validation function returned false.

3.5.7 Test Case 4.4

Test description: Input validation test with valid data for both start and end location
Result: Success.
Reason: The validation function returned true.

3.6 Unit Test Screenshot

3.7 Plans for the future

Once the application has been completed, we plan to conduct usability tests where we gather a number of people to test the application. The aim of these tests will be to determine whether the user interface of the application is user friendly and the test will also reveal any flaws in the system

4 Plan for Testing (Server)

4.1 getImages.py

To test the script, the following must be met

- Python version 2.7
- There must be a folder with the name "traffic_images"
- The computer where the script will be tested should have an internet connection
- Mysql server
- If Possible, you may verify if the itraffic website is operational.
- This script was written and tested on a UNIX environment and this is the environment where it should be tested.
- The "TrafficAnalyzer" Django project should be in the same folder as the "getImages.py". This is because the script makes use of the files in the Django project.

After the above requirements have been met,the script can be manually executed on the terminal or it may be executed.

4.2 Django project

The Django project will handle all the requests from the android device regarding camera information. The following is a list of the requirements for the Django application to work.

- Mysql server 5.6 or above
- Django 1.7.6
- python-mysql connector
- The project was developed in a UNIX environment.
- The server should have an internet connection.
- The port where the server is listening should be enabled. E.g 8000.

- The website also works with information form the itraffic website and thus it should be verified that the itraffic website is active to avoid avoid incorrect results from the server.

To Test the server, an Android device or a web browser may be used to send requests to the django application. Then the application should be able to respond with relevant information. Giving it known latitude-longitude combination and verify if the cameras returned are correct.