# Chapter 7

**Testing of the violation detection system**

During and after the implementation of the system, the system must be undergo continuous testing to ensure at every step that the system meets the requirements and needs of the user. After the implementation of the system is complete, it is necessary to test it in various scenarios to fix any bugs encountered during this process.

This process is called verification and validation of the system [40]. The primary purpose of testing is to identify bugs in the program and take corrective measures before it is ready for delivery as an application to the end user. It is an essential step in the process of finishing the system.

# Testing Environment

Testing was done on a Macbook Pro 2014 Model, connected to a power supply with the following specifications:

**Hardware Specifications:**

* 2.2GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.4GHz) with 6MB shared L3 cache12.00 GB RAM
* 64 bit OS
* 512 GB HardDisk

**Software Specifications:**

* Mac OSX Mavericks
* Vim editor, python2.7 and OpenCV 2.9

# Unit Testing of Main Modules

Unit testing (sometimes called component testing) is the process of testing individual components in the system [41]. This is a defect testing process so its goal is to expose faults in these components. In system testing, unit testing is a method by which individual modules of the whole system, sets of one or more modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use.Intuitively, one can view a unit as the smallest testable part of an application or system. Unit tests are created by the developers of the system or occasionally by white box testers during the development process.

* + 1. **Unit Testing of Background Modeling Module**

The Background Modeling unit is tested and test case is tabulated in the table. The test is performed to check whether the background modeling module is creating the scene background as specified by the rate of averaging of the video frames. The test was successful and the Table 7.1 references the tests:

**Table 7.1: Unit Test Case 1 to check if Background Modeling succeeds**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic violation detection system - 1 |
| Name of Test | Test to check if scene background is modeled convincingly. |
| Feature Being Tested | Create Scene Background. |
| Description | When the background modeling module is invoked, the video frames are averaged at a suitable rate to obtain the scene background. |
| Sample input | MPEG-4 format recording of vehicles jumping signal in a traffic scene. |
| Expected Output | Scene background with complete elimination of all foreground objects and incorporation of non-static background objects. |
| Actual Output | Scene background with complete elimination of all foreground objects and incorporation of non-static background objects. |
| Remarks | Test successfully completed. |

The figure 7.1 shows the scene with the foreground object. Figure 7.2 shows the object being averaged out of the scene. Finally, figure 7.3 shows the modeled background without any moving objects.

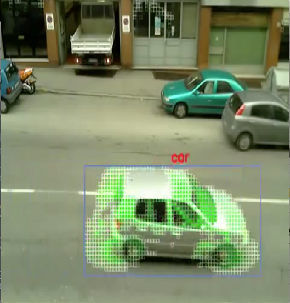


Figure 7.1: Scene with foreground object



Figure 7.2: Foreground object being removed



Figure 7.3: Modeled scene background

* + 1. **Unit Test of Foreground Modeling**

The Foreground modeling unit is tested and test case is tabulated in the table. The test conducted was to check if the foreground objects are successfully modeled in the scene. The input to this module is the video and the background from the background modeling module. The test was successful. The test is referenced in table 7.2

**Table 7.2: Unit Test Case 2 to check if the foreground modeling module succeeds**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic violation detection system-2 |
| Name of Test | Test to check if Foreground modeling succeeds |
| Feature Being Tested | Extraction of foreground objects in the scene |
| Description | The video frames are subtracted against the modeled background to obtain the foreground. |
| Sample input | MPEG-4 format traffic video and 3 channel background image |
| Expected Output | MPEG-4 format video with modeled foreground |
| Actual Output | MPEG-4 format video with modeled foreground |
| Remarks | Test Successfully completed. |

Figure 7.4 shows the original frame in the video and figure 7.5 shows the modeled foreground.

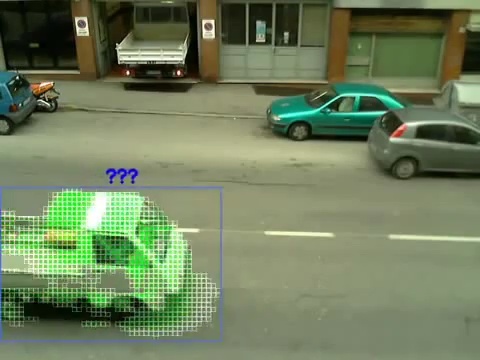


Figure 7.4: Original frame



Figure 7.5: Frame with modeled foreground

* + 1. **Unit test of Color quantization and background noise elimination**

In this unit test, the clustering based color quantization and background noise elimination module is tested and the test case is tabulated in the table. The test is takes input from the foreground modeling phase and performs clustering on them. The results of the phase are supposed to make the foreground object clearer and eliminate background noise. The test is referenced in Table 7.3.

**Table 7.3: Unit Test Case 3 check if Clustering and noise elimination succeeds**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic violation detection system-3. |
| Name of Test | Test to check if Clustering and noise elimination is achieved. |
| Feature Being Tested | Background noise elimination and foreground clarity. |
| Description | When the foreground frames are clustered the quantization of the image improves the clarity and eliminates all background noise. 2 clusters are used. |
| Sample input | MPEG-4 format video with modeled foreground with noise. |
| Expected Output | MPEG-4 format video with modeled foreground without noise and improved clarity. |
| Actual Output | MPEG-4 format video with modeled foreground without noise and improved clarity. |
| Remarks | Test completed successfully. |

Figure 7.5 shows the foreground modeled and figure 7.6 show the results of clustering.



Figure 7.5: Foreground modeled



Figure 7.6: Detailed foreground frame with no noise

* + 1. **Unit testing the Keyframe extraction module**

The Keyframe extraction module unit is tested and test case is tabulated in the table. The test conducted was to check whether redundant and blank frames are eliminated resulting in only frames that are useful in the next phase.. The test was successful. The test is referenced in table 7.4

**Table 7.4: Unit Test Case 4 to check if key frames are extracted**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic violation detection system-4. |
| Name of Test | Test to check if key frames are extracted. |
| Feature Being Tested | Blank and redundant frame elimination. |
| Description | The module uses the results of the clustering and filters the result using a mean squared error similarity checker. |
| Sample input | MPEG-4 format foreground modeled video after clustering |
| Expected Output | MPEG-4 format foreground modeled video with elimination of blank and redundant frames |
| Actual Output | MPEG-4 format foreground modeled video with elimination of blank and redundant frames |
| Remarks | Test Successfully Completed. |

Figure 7.7 shows the a few frames from the original foreground model with a blank frame between two frames of significant movement and figure 7.8 shows the result of key frame extraction.

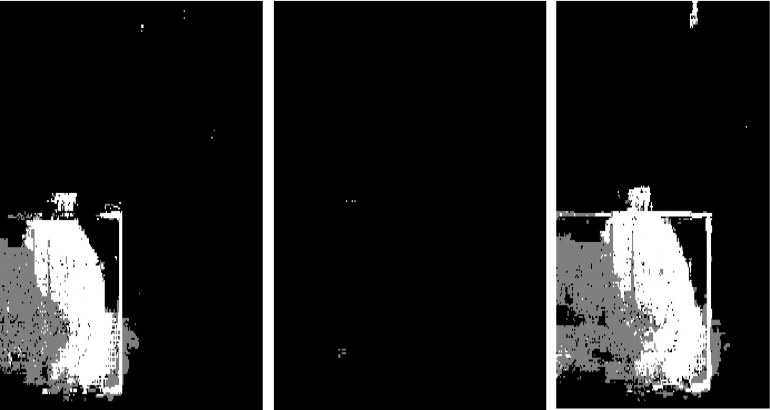


Figure 7.7: Frames with blank frame in between.

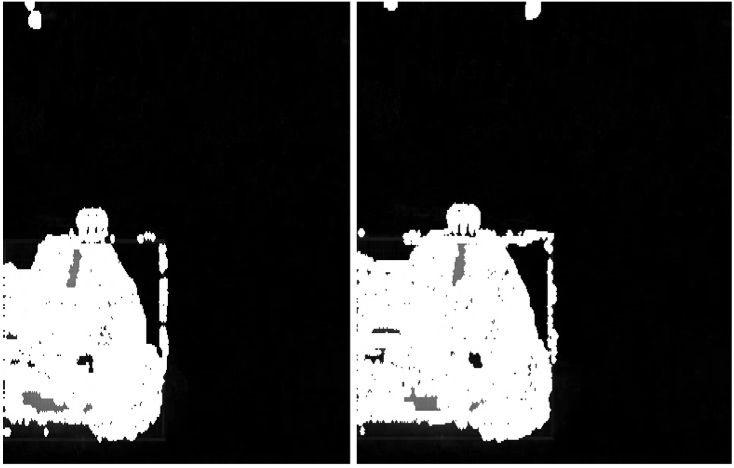


Figure 7.8: Key frames extracted.

* + 1. **Unit Testing of Violation detection Module.**

The Violation detection Module is tested and test case is tabulated in the table. The test conducted was to detect the vehicle crossing the edge of the frame, capturing the frame instance and time stamping it. The test was successful. The test is referenced in table 7.5

**Table 7.5: Unit Test Case 5 to check if violation detection is successful**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic violation detection system – 5. |
| Name of Test | Test to check if the violation is detected successfully. |
| Feature Being Tested | Capturing frame instance of video where violation occurs. |
| Description | When the vehicle in the frame crosses the edge of the frame, the frame is captured and time stamped. |
| Sample input | MPEG-4 format Recompiled video from the key frame extraction phase. |
| Expected Output | Captured frames showing the violation. |
| Actual Output | Captured frames showing the violation. |
| Remarks | Test Successfully Completed. |

Figure 7.9 shows the instance of violation captured and time stamped. Figure 7.10 shows violation captured with two vehicle objects in the scene. Figure 7.11 shows the violation captured with a two wheeler vehicle.



Figure 7.9: Violation Captured



Figure 7.10: Violation captured with two vehicle objects.



Figure 7.11: Violation captured with a two wheeler vehicle

* + 1. **Unit Testing of plate number recognition of violating vehicle.**

The Plate number recognition module is tested and test case is tabulated in the table. The test conducted was to check whether the number in the plate is recognized accurately by the digit classifier. The test was successful. The test is referenced in table 7.6

**Table 7.6: Unit Test Case 6 to check if number recognition succeeds**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic Violation Detection System – 6 |
| Name of Test | Test to check if the numbers on the plate are recognized |
| Feature Being Tested | Number recognition |
| Description | The numbers on the plate are recognized by a classifier. The classifier is trained initially using a digit data set and support vector machines. |
| Sample input | Plate extracted JPEG image |
| Expected Output | Number recognized as on plate |
| Actual Output | Number recognized as on plate |
| Remarks | Test Completed Successfully |

Figure 7.12 and figure 7.13 shows the numbers being successfully recognized. Figure 7.14 shows the number being partially recognized. The program incorporates an intelligent string matching algorithm to obtain the desired database filtering.



Figure 7.12: Number recognized.



Figure 7.13: Number recognized.



Figure 7.14: Number partially recognized.

* + 1. **Unit Testing of filtering database results module**

The Information filtering module is tested and test case is tabulated in the table. The test conducted was to check the number recognition results can successfully reduce the number of matched database results. The test was successful. The test is referenced in table 7.7

**Table 7.7: Unit Test Case 7 to check database filtering**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic Violation Detection System – 7. |
| Name of Test | Test to check if the database results are successfully filtered. |
| Feature Being Tested | Vehicle match from database results. |
| Description | The database is indexed by the recognized numbers and any match found is displayed. |
| Sample input | Text file containing the number recognized. |
| Expected Output | JPEG image displaying vehicles matched and their details. |
| Actual Output | JPEG image displaying vehicles matched and their details. |
| Remarks | Test Successfully Completed. |

Figure 7.15, figure 7.16 and figure 7.17 shows the results of database matching and vehicle details obtained. The vehicle information shown is extracted from the simulated vehicle database programmed manually in the system.

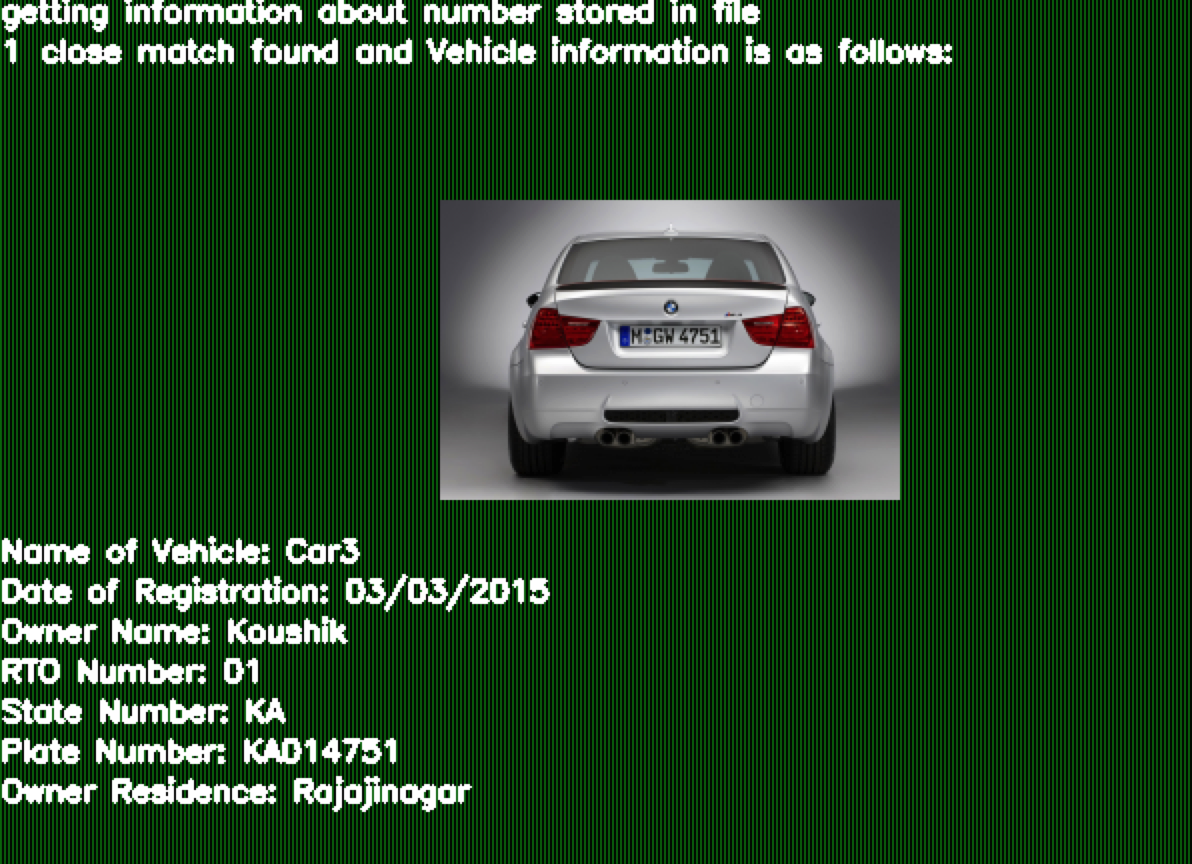


Figure 7.15: Database results.

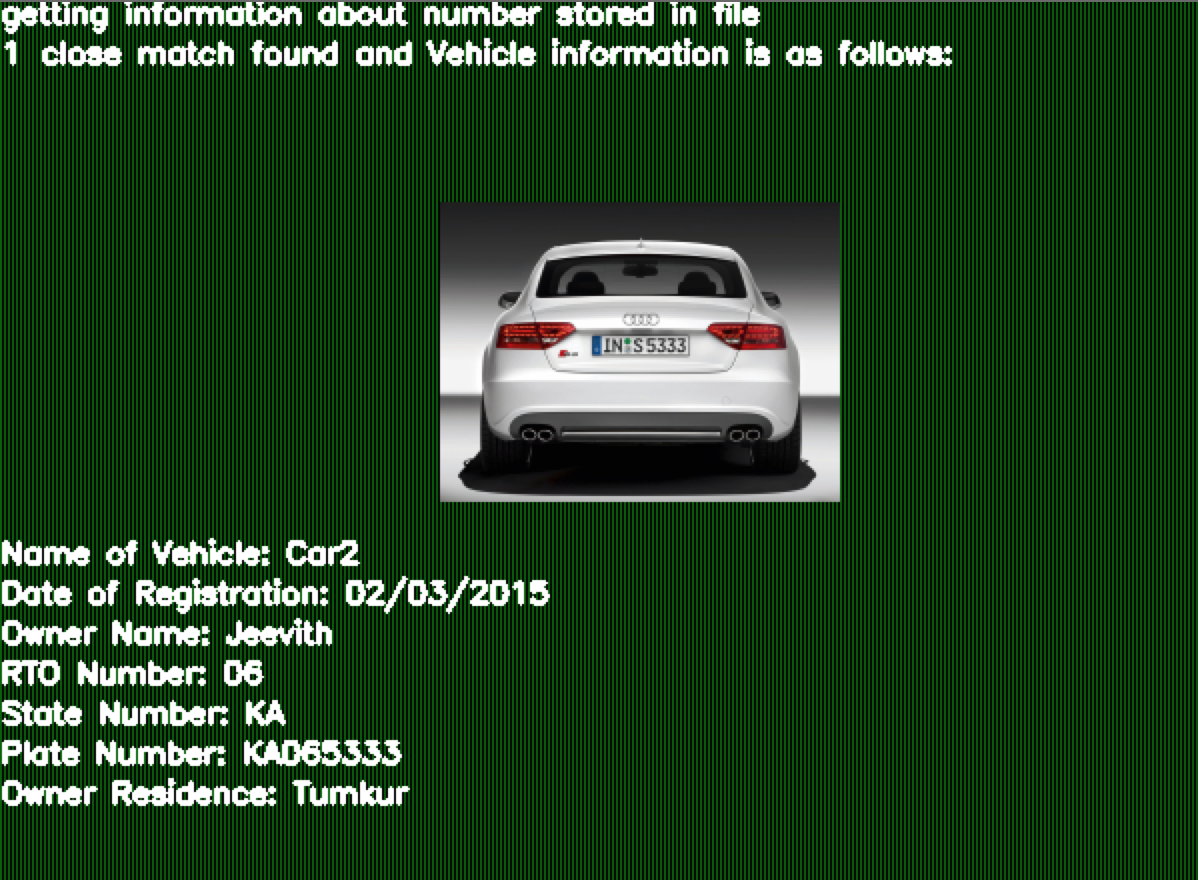


Figure 7.16: Database results.

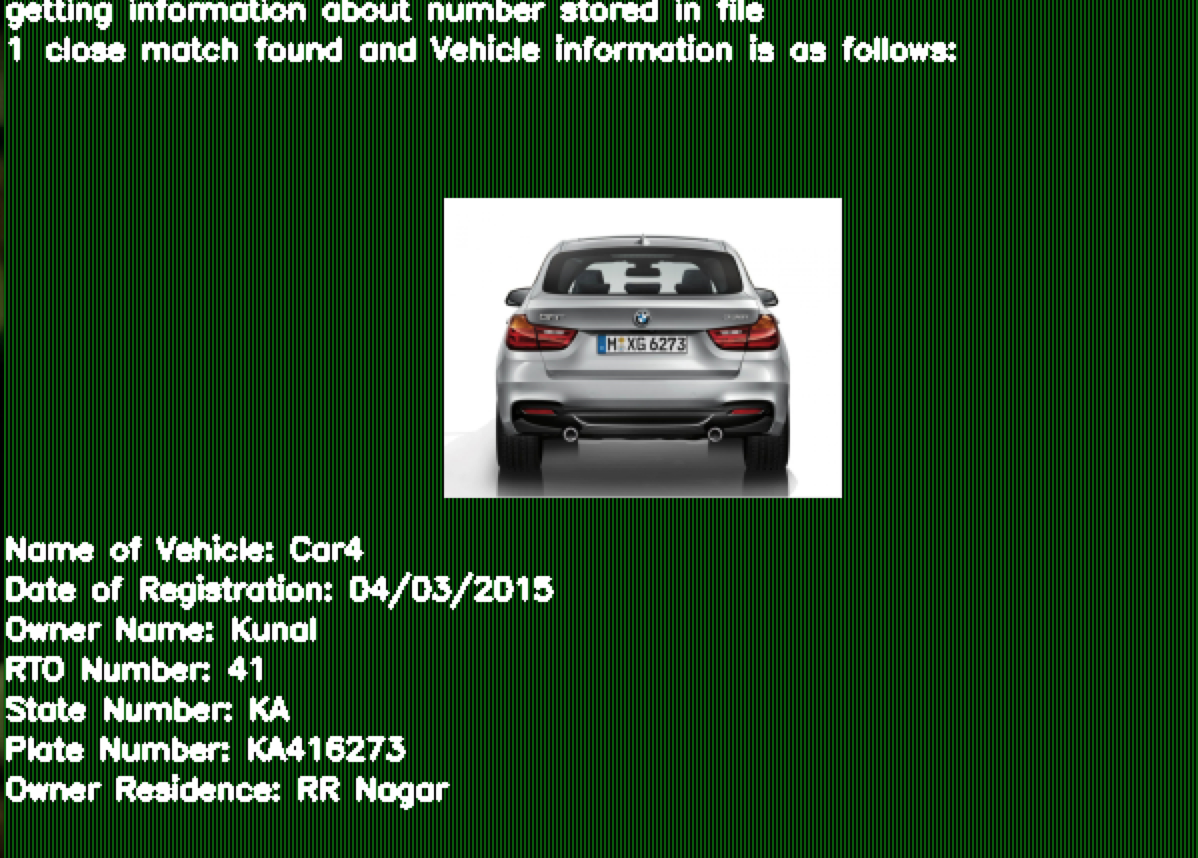


Figure 7.17: Database results.

# Integration Testing Of Modules

The process of system integration involves building a system from its components and testing the resultant system for problems that arise from component interactions [42]. The components that are integrated may be off-the-shelf components, reusable components that have been adapted for a particular system or newly developed components. For many large systems, all three types of components are likely to be used. Integration testing checks that these components actually work together are called correctly and transfer the right data at the right time across their interfaces.

In integrated testing, the modules are joined to form sub-systems which each perform a specific function. The performance of the product is determined by its holistic performance. The data selected for testing needs to carefully selected [43]. The test was conducted to see if the violations in the video and appropriate database results are displayed. This operation is one which creates a frame to capture the violation according to the input parameters and presents the number plate to the digit classifier. Next the information is filtered from the database using the plate numbers. This test was found to be successful. This is referenced in table 7.8:

**Table 7.8: Integrated Test Case 1 to Test case to see if Traffic violation detection system works.**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic Violation Detection System - 8 |
| Name of Test | To check if the violation results are obtained from the input video |
| Feature Being Tested | The functioning of all unit modules to obtain violation results |
| Description | All the modules are run serially to obtain the violation results as output |
| Sample input | MPEG-4 format traffic scene recording |
| Expected Output | Violation details from the database |
| Actual Output | Violation details from the database |
| Remarks | Test Successfully completed |

Intergration testing of the system is carried out to check if the all the modules work together and communicate the right information between the interfaces connecting the modules. The test was successful and the system works as required.

# System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. System testing fails within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. The entire Traffic Violation Detection system is tested from start to finish, including testing of all conditions and permutations. The test was found to be successful. The Test Case is tabulated in Table 7.9.

**Table 7.9: System testing of the Violation detection system**

|  |  |
| --- | --- |
| Sl No of Test Case | Traffic Violation Detection System -9. |
| Name of Test | System test. |
| Feature Being Tested | Traffic Violation Detection System. |
| Description | The flow of output between all the multiple stages of the system is tested with various types of input. The results obtained are checked for accuracy and consistency. |
| Sample input | Array details and input parameters to various operations. |
| Expected Output | All the required operations are performed successfully and the same is seen in the GUI of the vendors. |
| Actual Output | All the required operations are performed successfully and the same is seen in the GUI of the vendors. |
| Remarks | Test Successfully Completed. |

The system test is used to ensure the compliance of all modules of the system. It tests to see if the various modules work in synchronization with each other to provide the expected results. If the test is a success, it indicates system is ready for use.

# Summary

This chapter has described the various unit, integration and system tests conducted on the Traffic Violation Detection System. The input parameters, output results and description of each test is mentioned in the table representing the test. Finally, all tests have been passed and the system is ready for use.