# Chapter 6

# Testing

Testing is an important phase in the development life cycle of the product. In this phase, the error from all the phases was detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. During testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program is performing as expected. Errors were found and corrected by using the following testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system before it was ready for implementation.

Software Testingis the process used to help identify the correctness, completeness, security, and quality of the developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, which is intended to reveal quality related information about the product with respect to the context in which it is intended to operate.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations that the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester.

**6.1 Test Environment**

The testing of the modules was done on machine with the following features.

* Operating System : Windows 7
* JDK Version : 1.7
* Browser : Google Chrome
* Web server : Tomcat Version 7

**6.2 Unit Testing**

The features to be tested are the individual modules of the system developed for the correctness of their functionality. So the items under consideration can be said as all the major modules of the system. The main purpose of Unit testing is to remove any irregularities present in the individual modules of the system. Any errors in the output of the modules are also repaired. The criteria based on which the modules are tested are the correct functioning of the module. Deviating from the basic functionality can lead to errors in the expected output in which case the module is considered to fail under those circumstances.

In computer programming, unit testingis a procedure used to validate whether individual units of source code are working properly. A unit is the smallest entity of an application. An input is provided to the module which then operates accordingly and gives the calculated output. This output is matched with the expected output and then the decision is taken whether the module is working as expected. Here it is required to validate the user input.

**Testing strategy**

To test the modules involved in our project Security in audio Steganography, the approach we follow is the manual unit testing approach. Usually a debugging involves providing a range of inputs which include both valid and invalid inputs and checking the response of each unit. Advantage of manual approach is that it is highly visible to current developer. Each test case includes many features describing the input output transformations. The features to be tested include the accuracy of the individual unit and also the range of input for which the unit functions properly.

The purpose of the testing is as follows.

**Test Cases**

The details of the individual test cases listed as follows

**Module 1: Access data trough Twitter APIs**

If user is entered a keyword using Twitter APIs program has to retrieve the tweets.

**Table 6.1: Test Case-1**

|  |  |
| --- | --- |
| Sl. No. of test case : | 01 |
| Name of test : | Accessing Twitter Data |
| Item / Feature being tested : | Twitter APIs. |
| Description : | In this project Twitter4j an Open Source package is used to retrieve the Twitter data. |
| Sample Input : | Keyword to the search API. |
| Expected output : | Tweets containing that keyword |
| Actual output : | Tweets containing that keyword |
| Remarks : | API works successfully as expected. |

**Module 2: Preprocessing**

In this module pre processing(such as replacing @username by AT\_USERNAME, #content by content, removing of characters such as ? \* etc) test is done. Pre processed twitter data is obtained.

**Table 6.2: Test Case-2**

|  |  |
| --- | --- |
| Sl. No. of test case : | 02 |
| Name of test : | Preprocessing |
| Item / Feature being tested : | Tweets |
| Description : | This module helps removing retweets, usernames and url in the tweets. |
| Sample Input : | Tweets retrieved for particular keyword  For example: RT @sachin rvce reallllly rocks [www.rvce.edu.in](http://www.rvce.edu.in) |
| Expected output : | A file containing tweets pre processed  For example: USERNAME rvce realy rocks URL |
| Actual output : | A file containing tweets pre processed  For example: USERNAME rvce realy rocks URL |
| Remarks : | Module works successfully as expected. |

**Module 3:**

**Table 6.3: Test Case-3**

|  |  |
| --- | --- |
| Sl. No. of test case : | 03 |
| Name of test : | Working of Google Chart API |
| Item / Feature being tested : | Charts |
| Description : | Google is giving support to draw pie charts and graphs using Google APIs. This module helps to test those APIs. |
| Sample Input : | Data in the form of integer array |
| Expected output : | Pi Chart |
| Actual output : | Chart is plotted |
| Remarks : | Test case passed |

**Module 4: Data file for reading and writing**

**Table 6.4: Test Case-4**

|  |  |
| --- | --- |
| Sl. No. of test case : | 04 |
| Name of test : | Data file reading and writing |
| Item / Feature being tested : | File |
| Description : | In this project reading and writing is done for training data and twitter data files so it is necessary to check weather files are presents in that location. |
| Sample Input : | File path |
| Expected output : | File found at specified location |
| Actual output : | File found at specified location |
| Remarks : | Module works successfully as expected. |

**Module 4: Naïve Bayes algorithm testing**

**Table 6.5: Test Case-5**

|  |  |
| --- | --- |
| Sl. No. of test case : | 05 |
| Name of test : | Naïve Bayes |
| Item / Feature being tested : | Tweets |
| Description : | Naïve Bayes algorithm takes word count of training data, training data, input file containing tweets and compares the data. It calculates the probability of occurrence positive, negative and neutral sentiments. |
| Sample Input : | Training data and Test data |
| Expected output : | Classified Tweets |
| Actual output : | Classified Tweets |
| Remarks : | Module works successfully as expected. |

**6.3 Integration Testing**

Integration testing is a level in software testing in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until software works as a whole.

Checking of different components is done to see whether they interact with each other and work in an expected manner. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

**Testing strategy**

**Purpose:** The system is designed to work without the intervention of the user. The user enters only the keyword (based on which tweets are retrieved). Once the input is received, it is the job of the application to classify the tweets accurately. After unit testing, all the modules are integrated to check whether they work as expected.

**Table 6.6: Test Case-6**

|  |  |
| --- | --- |
| Sl No. of test case : | 06 |
| Name of test : | Integration of all modules |
| Item / Feature being tested : | Check for classification of the retrieved tweets |
| Sample Input : | Input in the form @username  For example. @barackobama |
| Description: | This test is done in following steps:   * Using Twitter4j search API application search for the tweets containing that @keyword. * Pre-processing of retrieved tweets takes place. * Sentiment Analysis done by Naïve Bayes approach. * All the tweets are classified as positive, negative and neutral. |
| Expected output : | Tweets retrieved and classified. |
| Actual output : | Same as above |
| Remarks : | Test passed |

**Table 6.7: Test Case-7**

|  |  |
| --- | --- |
| Sl No. of test case : | 07 |
| Name of test : | Integration of all modules |
| Item / Feature being tested : | Check for classification of the retrieved tweets |
| Description: | This test is done in following steps:   * Using Twitter4j search API application search for the tweets containing that #keyword. * Pre-processing of retrieved tweets takes place. * Sentiment Analysis done by Naïve Bayes approach.   All the tweets are classified as positive, negative and neutral. |
| Sample Input : | Input in the form #subject. For example #ipl. |
| Expected output : | Tweets retrieved and classified |
| Actual output : | Same as above |
| Remarks : | Test passed |

**6.4 System Testing**

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

**Table 6.8: Test Case-8**

|  |  |
| --- | --- |
| Sl. No. of test case : | 08 |
| Name of test : | Integration of all software modules |
| Item / Feature being tested : | The system works as a whole |
| Description: | Check whether the project code work as a whole and produce results as expected |
| Sample Input : | Input in the form @username or #subject |
| Expected output : | Tweets retrieved and classified and written to an appropriate file |
| Actual output : | Tweets retrieved and classified and written to an appropriate file |
| Remarks : | Test case passed |

**6.5 User Interface Testing**

Interface is actually software that consists of sets of messages, commands, images, and other features that allow communication between a device and a user. Here the interface testing mainly focuses on testing the user interface.

**Test Strategy**

The interface testing done focuses on the user interface provided. Unlike other modules testing of user interface is different in the sense many of the testing scenarios are subjective. That is there is no clear cut definition as to whether the feature stated here are actually satisfied in our project some of the necessary features of a good user interface are as follows.

* User familiarity: the interface should use terms and concept, which are drawn from the experience of the people who will use the system.
* Consistency: The interface should be consistent in that, wherever possible comparable operations should be activated in the same way.
* Recoverability: The interface should include mechanisms to allow users to recover from errors.
* User guidance: The interface should provide meaningful feedback to allow the users to recover from errors.

**User Interface Test Cases**

**Module 1: Test whether the user have entered correct keyword.**

**Table 6.9: Test Case-9**

|  |  |
| --- | --- |
| Sl. No. of test case : | 9 |
| Name of test : | Checking for the keyword given in the input box is valid or not. |
| Item / Feature being tested : | Keyword |
| Description : | User is supposed to enter either alphanumeric keyword or completely alphabetic keyword, but not completely numeric. |
| Sample Input : | Keyword containing only numbers. For example #33333,@123 etc |
| Expected output : | Message - “Enter either alphanumeric keyword or completely alphabetic keyword” is displayed. |
| Actual output : | Message - “Enter either alphanumeric keyword or completely alphabetic keyword” is displayed. |
| Remarks : | Module works successfully as expected. |

**Module2: Test whether the Apache server is running or not.**

Access to GUI without running the Apache server will display an error.

**Table 6.10: Test Case-10**

|  |  |
| --- | --- |
| Sl. No. of test case : | 10 |
| Name of test : | Checking for the Access to GUI without running the Apache |
| Item / Feature being tested : | Running of the Apache |
| Description : | Apache is of primal importance for working of the project. When GUI is accessed without running this server an error is displayed. |
| Sample Input : | GUI |
| Expected output : | Server is not running |
| Actual output : | Server is not running |
| Remarks : | Module works successfully as expected. |

**Module3: Checking for number of tweets available**

Test to check what happens if user enters keyword that contains less tweets than the user specified.

**Table 6.11: Test Case-11**

|  |  |
| --- | --- |
| Sl. No. of test case : | 11 |
| Name of test : | Checking for number of tweets available |
| Item / Feature being tested : | Tweet count |
| Description : | When user enters the keyword which does not have the specified number of tweets an error message is displayed. |
| Sample Input : | Tweet count and keyword |
| Expected output : | “Cannot able to retrieve more tweets for this keyword” message is displayed. |
| Actual output : | “Cannot able to retrieve more tweets for this keyword” message is displayed. |
| Remarks : | Module works successfully as expected. |