CHAPTER 7: System Testing

System testing is conducted on a complete, integrated system to evaluate the system’s fulfillment with its specified requirements. System testing is a form of validation or verification for the system upon the system’s completion. It involves the review of the specification, design and coding. This phase is crucial to check the system quality control and assurance. There are few rules that suitable for system testing objectives and they have been used to test the application. The rules are;

- Testing is a process of which programs is executed with precise plans to find errors and run-time program bugs
- An effective test case is one which contains unexpected testing records sets with high probability of detecting undiscovered errors during the program design and development phases
- A successful test is also not one which uncovers only few expected errors; it is one which constantly provides new challenges to its programmer over time.

7.1 Testing Techniques

The testing technique that is used in testing the application is focused in two types of testing which are white box and black box.
7.1.1 White-box Testing

White box testing or clear box testing used internal viewpoint of the system to design test case based on internal structure. This testing requires programming ability to recognize all paths through the system. Tester should choose test case inputs to carry out paths through the codes and verify appropriate outputs. The application system has been tested using the white box texting where it is approved to have suitable paths and proposed outputs.

White box testing is based on an actual implementation, if any implementation changes then the test probably have to change as well. When the application needs some changes and alterations, another ways of implementation also is planned and the manner continued until the Image Encryption’s application meets the requirements of the test.

White box testing is applicable at the unit, integration and system levels of the system testing process. Nonetheless this testing technique of test design can reveal a tremendous number of test cases. It might not discover unimplemented parts of the specification or missing requirements, but it is sure that all paths through the test object are executed.

There few standard white box testing design:

- Control flow testing
- Data flow testing
- Branch Testing

In meeting all of the three tests, the application has not revealed a tremendous number of test cases like it should be. This can be related to the use of Waterfall model earlier which has benefited the testing process much.
7.1.2 Black-box Testing

Unlike white box testing, black box testing takes an external viewpoint of the test object to derive test cases. Tests can be functional or non-functional but usually it is functional. In this case, the test also is functional.

In Black-box testing, tester chooses valid and invalid inputs and decides the correct outputs. There is absolutely zero knowledge of the test object’s internal structure. The application has been tested by choosing the valid inputs and later by selecting the invalid inputs. The image is uploaded and keyed in with a password to encrypt. However, to decrypt the image, a different, invalid password has been used to receive the image. As the outcome result is zero knowledge to the tester, it gave a warning of wrong password to the tester.

Black box testing method of design testing is applicable to all stages of software testing as it has proved in the thesis application. The advanced the stage, the larger and more complicated the box, the more it is to use black box testing to simplified it. While this method can reveal unimplemented parts of specification, it can’t ever be sure that all existent paths are tested.

Common types of black box testing:

- Equivalence partitioning
- Boundary value analysis
- Decision table testing
- Pair wise testing
- State transition tables
- Use case testing
- Cross-functional testing
However, in the thesis project the parts that need to be tested are the encryption and decryption of the image and the scrambled image. Thus, it is only relevance to the equivalence partitioning, boundary value analysis and use case testing. This is because the retrieval of the image and the outcome of the encryption are much dependent on the data’s value.

7.2 Testing Strategies

There is going to be a few testing that is going to be done in the application. The testing strategies such as unit, integration and system are done to test the system’s reliability.

7.2.1 Unit Testing

Unit testing is a method used to validate specific component of the coding if it is working correctly. This test is only done by the developer not the users. The objective of unit testing is to separate each component of the program and to prove which individual component is correct.

Each module in the application holds sub-modules and sub-modules holds functions. Functions or components are individually tested before the entire module is tested. In the development of the system, unit testing was performed after development of each of the component and it is a non-stop process throughout the coding phase.

There some unit testing being done in the application that will be shown in the table below:
<table>
<thead>
<tr>
<th>Test Procedure</th>
<th>Expected Output</th>
<th>Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading the image</td>
<td>Image is loaded and previewed</td>
<td>Successful</td>
</tr>
<tr>
<td>Image processed is encrypted</td>
<td>Encrypted image will be show in jumble form</td>
<td>Successful</td>
</tr>
<tr>
<td>Password hash is embedded in the image</td>
<td>Password hash is saved in the end of file</td>
<td>Successful</td>
</tr>
<tr>
<td>Incorrect password is entered</td>
<td>Error alert will be shown</td>
<td>Successful</td>
</tr>
<tr>
<td>Encrypted image confidentiality</td>
<td>Receiver cant retrieve the image if incorrect password is entered</td>
<td>Successful</td>
</tr>
<tr>
<td>Message receive is decrypted</td>
<td>Original image is shown with correct password</td>
<td>Successful</td>
</tr>
</tbody>
</table>

Table 7.1 Unit Testing Done

### 7.2.2 Integration Testing

Integration testing obtains inputs modules that have been unit tested, collects those modules in a bigger collection, applies tests and convey the output to the integrated system that it is ready for testing.

The objective of this integration testing is to prove functional, performance and reliability requirements placed on main design items. These design items are employs through their interfaces using Black box testing, success or error circumstances being simulated via appropriate parameter and data inputs.

The overall idea is a "building block" method, in which verified collections are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up are not suitable to test the image encryption application because they did not share certain significant method to prove the functionality of the application.
7.2.3 System Testing

System testing is intended to test up to and beyond the bounds defined in the software/hardware requirements specifications as we discuss in Chapter 4.