**Steganography Project Master Test Plan**

Linden Crandall, Jonathan Mainhart, Zhihua Zheng

University of Maryland Global Campus

CMIS 495: Current Trends and Projects in Computer Science

Prof. Majid Shaalan

April 5, 2022

**Introduction**

The tests listed in this document will be used to measure the functionality of the application. Test tables are organized according to functional area. Table 1 contains GUI tests, table 2 contains encoding and decoding tests, table 3 contains file I/O tests, and integration tests are contained in table 4.

**Test Cases**

**Table 1**

GUI Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| 1 | What is being tested | How it will be tested | What you expect to see |
|  |  |  |  |

**Table 2**

Encoding and Decoding Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| 1 | pixel extractor | 300 x 300 all black jpeg | [(0,0,0), (0,0,0), (0,0,0) … (0,0,0)] (90,000 black pixels) |
| 2 | message encoder | “The quick brown fox jumps over the lazy dog” | 01010100 01101000 01100101 00100000 01110001 01110101 01101001 01100011 01101011 00100000 01100010 01110010 01101111 01110111 01101110 00100000 01100110 01101111 01111000 00100000 01101010 01110101 01101101 01110000 01110011 00100000 01101111 01110110 01100101 01110010 00100000 01110100 01101000 01100101 00100000 01101100 01100001 01111010 01111001 00100000 01100100 01101111 01100111 |
| 3 | message decoder | 01010100 01101000 01100101 00100000 01110001 01110101 01101001 01100011 01101011 00100000 01100010 01110010 01101111 01110111 01101110 00100000 01100110 01101111 01111000 00100000 01101010 01110101 01101101 01110000 01110011 00100000 01101111 01110110 01100101 01110010 00100000 01110100 01101000 01100101 00100000 01101100 01100001 01111010 01111001 00100000 01100100 01101111 01100111 | “The quick brown fox jumps over the lazy dog” |
| 4 | image encoder | “The quick brown fox jumps over the lazy dog”  300 x 300 all black jpeg | [(0,1,1), (0,1,1), (0,1,0), (0,1,1), (1,0,0), (0,0,0) … (0,1,1), (0,0,1), (1,1,1) (pixel bits shifted odd for 1, even for 0 which will match the pattern of the binary encoded message with an additional even bit after each character if there are more characters to follow or an odd bit if it is the end of the message) |
| 5 | image decoder | 300 x 300 all black jpeg encoded with “The quick brown fox jumps over the lazy dog” | “The quick brown fox jumps over the lazy dog” |
| 6 | maximum character length calculator | 300 x 300 all black jpeg | 33,750 characters (300 pixels x 300 pixels x 3 bits per pixel/8 bits per character) |

**Table 3**

File I/O Tests – OPENING/READING/WRITING/CLOSING IMAGE FILE

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| 1 | image file chooser | No input, no parameters | - User’s default file explorer opened  - user selects file  - if the file is an image file (.png, .jpg, etc.), assign the image file to an image object variable in code for further processing  - else throw exception (ex. dialog pop up) |
| 2 | save image | image object | - user’s default file explorer is opened  - save image to chosen path  - throw exception if image is not properly saved (ex. user exits out of application while in the file explorer) |

**Table 4**

Integration Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| 1 | What is being tested | How it will be tested | What you expect to see |
|  |  |  |  |