**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, table 4 contains reset image tests, and integration tests are contained in table 5.

**Test Cases**

**Table 1**

Automated Unit Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Test Input** | **Expected** | **Actual** | **Pass/Fail** | **Fig.** |
| 1 | check max chars of 300x300 pix img | 30,000 | As expected | Pass |  |
| 2 | check max chars of 5x5 pix img | 8 | As expected | Pass |  |
| 3 | encode 5x5 pix img to max chars without truncation | ‘01234567’ | As expected | Pass |  |
| 4 | encode 5x5 pix image to max chars with overflow input ‘01234567ABCDEF’ | ‘01234567’ | As expected | Pass |  |
| 5 | image and backup image attributes are exact copies | Attributes match | As expected | Pass |  |
| 6 | image encode changes pixels to not match backup image | Attributes do not match | As expected | Pass |  |
| 7 | check decoded message matches encoded message | Messages match | As expected | Pass |  |
| 8 | check the image can save to disk as a new file | new file saved | As expected | Pass |  |
| 9 | check the saved image decoded message matches the encoded message | Messages match | As expected | Pass |  |
| 10 | check that encoded image pixel values match original values after reset | Images reset | As expected | Pass |  |
| 11 | check convert message utility return value matches binary representation of test message | Values match | As expected | Pass |  |
| 12 | File input exception |  |  |  |  |
| 13 | File output exception |  |  |  |  |
| 14 | File overwrite exception |  |  |  |  |
| 15 |  |  |  |  |  |

**Table 2**

GUI Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Test Input** | **Expected** | **Actual** | **Pass/Fail** | **Fig.** |
| 1 | Randomly selected image loads on launch | Image displays | As expected | Pass |  |
| 2 | Image is decoded when loaded | Decoded message displays | As expected | Pass |  |
| 3 | Open image button opens file chooser | File chooser displays | As expected | Pass |  |
| 4 | File chooser has read access to user file system | Able to navigate file system | As expected | Pass |  |
| 5 | Attempt to open non-image file | File chooser rejects non-image files | As expected | Pass |  |
| 6 | Attempt to open image file | Image opens and is displayed | As expected | Pass |  |
| 7 | Attempt to enter test phrase into text input area | Text is displayed. Allowed character count decreases | As expected | Pass |  |
| 8 | Attempt to enter text greater than maximum allowed for the image | Text box does not allow input beyond maximum. Warning message displays | As expected | Pass |  |
| 9 | Encode image button pressed | Text input is disabled | As expected | Pass |  |
| 10 | Save image button pressed | Save file dialog opens | As expected | Pass |  |
| 11 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table 1**

GUI Tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected** | **Actual** | **Pass / Fail** | **Fig.** |
| 1 | The title of the application | “Steganosaurus” | Displays “Steganosaurus” |  |  |  |
| ~~2~~ | ~~Image panel~~ | ~~N/A~~ | ~~Displays nothing~~ |  |  |  |
| ~~3~~ | ~~Image panel~~ | ~~A valid image is uploaded~~ | ~~Displays the uploaded image~~ |  |  |  |
| ~~4~~ | ~~Image panel~~ | ~~An invalid image is uploaded~~ | ~~Displays nothing~~ |  |  |  |
| ~~5~~ | ~~TextField~~ | ~~N/A~~ | ~~Displays nothing~~ |  |  |  |
| 6 | TextField | * No image is displayed * TextField input: “abcABC123#$%...” | Displays “abcABC123#$%...” |  |  |  |
| 7 | TextField | * Image is displayed * TextField input: N/A | Displays message “Remaining character is ‘number’.” |  |  |  |
| 8 | TextField | * Image is displayed * Remaining characters number is greater than or equal to 0 | Displays ‘TextField input’ |  |  |  |
| 9 | TextField | * Image is displayed * Remaining characters number is less than 0 | Displays warning message “Invalid input. Exceeded the maximum capacity by ‘number’ characters.” |  |  |  |
| ~~10~~ | ~~“Open Image” button~~ | ~~“Open Image”~~ | * ~~Displays “Open Image”~~ * ~~Button name is centered~~ * ~~Button is enabled~~ |  |  |  |
| 11 | “Save Image” button | “Save Image” | * Displays “Save Image” * Button name is centered * Button is enabled |  |  |  |
| 12 | “Encode Data” button | “Encode Data” | * Displays “Encode Data” * Button name is centered * Button is enabled |  |  |  |
| 13 | “Decode Data” button | “Decode Data” | * Displays “Decode Data” * Button name is centered * Button is enabled |  |  |  |
| 14 | “Reset Image” button | “Reset Image” | * Displays “Reset Image” * Button name is centered * Button is enabled |  |  |  |

**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 | Choose Image | sample\_image.jpg | * User’s default file explorer opened * user selects sample\_image.jpg * continue (make a copy of image for resetting purposes and store both images in code for further processing, etc.) |
| 2 | Choose Image | invalid\_file.txt | * user’s default file explorer is opened * user selects invalid\_file.txt * throw exception thrown for invalid file type (ex. dialog pop up) * return to caller |
| ~~3~~ | ~~Save image~~ | ~~sample\_image.jpg~~ | * ~~save image to chosen path~~ * ~~throw exception if image is not properly saved (ex. user exits out of application while in the file explorer)~~ |
| ~~4~~ | ~~Save image~~ | ~~No image previously chosen or stored in code~~ | * ~~throw exception thrown for nonexistent image file (ex. dialog pop up)~~ * ~~return to caller~~ |

**Table 4**

Reset Image Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| ~~1~~ | ~~Reset image~~ | ~~N/A~~ | ~~Do nothing~~ |
| ~~2~~ | ~~Reset image~~ | * ~~A valid image is uploaded~~ * ~~Data is not encoded~~ * ~~Data is not decoded~~ | ~~Do nothing~~ |
| ~~3~~ | ~~Reset image~~ | * ~~A valid image is uploaded~~ * ~~Data is encoded~~ | * ~~Data variable equals an empty string~~ * ~~Discard encoded image. Retrieve and return previously stored copy~~ |
| ~~4~~ | ~~Reset image~~ | * ~~A valid image is uploaded~~ * ~~Data is decoded~~ | ~~Do nothing~~ |

**Table 5**

Integration Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Function** | **Test Input** | **Expected Output** |
| 1 | * Open Image * Encode Image * Save Image | * A valid image is selected * TextField input: Remaining characters number is greater than or equal to 0 | * User’s default file explorer opened * GUI displays selected image * An image with the encoded data is generated * A dialog pops up with the message “Successfully encoded the message!” * The image with the encoded data is saved to the chosen path * The original image is deleted * No error occurs throughout the test |
| 2 | * Open Image * Decode Image | * A valid image with the encoded data is selected   (The image generated from test 1 can be used) | * User’s default file explorer opened * GUI displays selected image * A dialog pops up with the message “Successfully decoded the message!” * TextField displays the decoded message * No error occurs throughout the test |
| 3 | * Open Image * Encode Image * Reset Image | * A valid image is selected * TextField input: Remaining characters number is greater than or equal to 0 | * User’s default file explorer opened * GUI displays selected image * An image with the encoded data is generated * A dialog pops up with the message “Successfully encoded the message!” * The image with the encoded data is deleted * The original image is not deleted * No error occurs throughout the test |
| 4 | * Open Image * Encode Image * Save Image | * Initially an invalid image is selected, then select a valid image to upload * Initial enter an invalid TextField input, then enter a valid TextField input | * User’s default file explorer opened * A dialog pops up with the warning message “Failed to open the image, please check the image file.” * GUI displays selected image * Displays warning message “Invalid input. Exceeded the maximum capacity by ‘number’ characters.” * An image with the encoded data is generated * A dialog pops up with the message “Successfully encoded the message!” * The image with the encoded data is deleted * The original image is not deleted |