**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 29, 2022

# Introduction

This document contains test cases used to measure functionality of the application. The tests were run on four different systems listed in table 1. Automated unit test cases for the backend (application logic) and frontend (user interface) are contained in table 2 and table 3 respectively. Manual integration tests are listed in table 4.

Descriptions and screen shots of the results of each test begin on page x.

# Test Systems

The application was tested on four separate systems listed in table 1. These systems represent the expected typical application user systems. Each system has Python version 3.9 installed. A Linux system was unavailable at the time of testing.

**Table 1**

Test Systems

|  |  |  |  |
| --- | --- | --- | --- |
| System ID | OS (Version) | Processor | RAM |
| S1 | macOS (12.3.1) | 3.8 GHz 8-Core Intel i7 | 8 GB 2667 MHz DDR4 |
| S2 | macOS (12.1) | 1.6 GHz Dual-Core Intel i5 | 8 GB 2133 MHz LPDDR3 |
| S3 | Windows |  |  |
| S4 | Windows |  |  |

# Test Cases

**Table 2**

Automated Tests (Back End)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Test Input | Expected | Pass/Fail | | | | Fig. # |
| S1 | S2 | S3 | S4 |
| 1 | check max chars of 300x300 pix img | 30,000 | Pass | Pass |  |  |  |
| 2 | check max chars of 5x5 pix img | 8 | Pass | Pass |  |  |  |
| 3 | encode 5x5 pix img to max chars without truncation | ‘01234567’ | Pass | Pass |  |  |  |
| 4 | encode 5x5 pix image to max chars with overflow input ‘01234567ABCDEF’ | ‘01234567’ | Pass | Pass |  |  |  |
| 5 | image and backup image attributes are exact copies | Attributes match | Pass | Pass |  |  |  |
| 6 | image encode changes pixels to not match backup image | Attributes do not match | Pass | Pass |  |  |  |
| 7 | check decoded message matches encoded message | Messages match | Pass | Pass |  |  |  |
| 8 | check the image can save to disk as a new file | new file saved | Pass | Pass |  |  |  |
| 9 | check the saved image decoded message matches the encoded message | Messages match | Pass | Pass |  |  |  |
| 10 | check that encoded image pixel values match original values after reset | Images reset | Pass | Pass |  |  |  |
| 11 | check convert message utility return value matches binary representation of test message | Values match | Pass | Pass |  |  |  |

**Table 3**

Automated Unit Tests (Front End)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Test Input | Expected | Pass/Fail | | | | Fig. # |
| S1 | S2 | S3 | S4 |
| 1 | on\_open\_button\_click | ImageObject matches. |  | Pass |  |  |  |
| 2 | on\_encode\_button\_click.  class variable: enable\_bool = True. | * Object display\_image’s method encode\_image is called once with default text from TextField. * update\_widgets\_status is called with the arguments: (False, True, False). |  | Pass |  |  |  |
| 3 | on\_encode\_button\_click.  class variable: enable\_bool = False. | * Method popip\_user\_notification is called with the arguments: (‘Failed to execute encode function! \nPlease modify the text field input.', MainWidget.MESSAGE\_TYPE.ERROR) |  | Pass |  |  |  |
| 4 | on\_reset\_button\_click | * Variable warning\_type maches MainWidget.WARNING\_TYPE.RESET * Method popup\_user\_notification is called with the arguments: (‘Are you sure you want to reset the image?', MainWidget.MESSAGE\_TYPE.WARNING) |  | Pass |  |  |  |
| 5 | execute\_reset | * Object display\_image’s method reset\_image is called once. |  | Pass |  |  |  |
| 6 | on\_save\_button\_clic | * Variable new\_filename maches the String “expected”. * ImageChooserPopup Class method show\_filechooser is called once. |  | Pass |  |  |  |
| 7 | on\_save  setup: new\_filename is valid.  overwrite with new\_filename. | * Method popup\_user\_notification is called with the arguments: (‘Image name already exists. \nAre you sure you want to overwrite the image?', MainWidget.MESSAGE\_TYPE.WARNING) |  | Pass |  |  |  |
| 8 | execute\_save  setup:  decode\_image method return string “decoded\_msg”  new\_filename = 'test\_image\_3.jpeg'  new\_filepath = 'path' | * Object display\_image’s method save\_image is called once with arguments: ('path', 'test\_image\_3.jpeg') * Variable title maches the String “Steganosaurus – test\_image\_3.jpeg”. * main\_image.source maches the String “path/ test\_image\_3.jpeg”. * Variable textfield\_str matches “decoded\_msg”. |  | Pass |  |  |  |
| 9 | validate\_image\_name  setup: image\_name = ‘test\_image\_3.jpeg' | * Method validate\_image\_name returns True. |  | Pass |  |  |  |
| 10 | validate\_image\_name  setup: image\_name = ‘\_test\_image\_3.jpeg' | * Method validate\_image\_name returns False. * Method popup\_user\_notification is called with the arguments: (‘Invalid file name! \n Only alphabet characters, numbers, dot, underscore and hyphens are allowed. (e.g. image\_1)', MainWidget.MESSAGE\_TYPE.ERROR) |  | Pass |  |  |  |
| 11 | validate\_image\_name  setup: image\_name = ‘\_test\_image\_3.jpeg' | * Method validate\_image\_name returns False. * Method popup\_user\_notification is called with the arguments: (‘Invalid file name! \n Only alphabet characters, numbers, dot, underscore and hyphens are allowed. (e.g. image\_1)', MainWidget.MESSAGE\_TYPE.ERROR) |  | Pass |  |  |  |
| 12 | update\_warning\_btn\_yes  setup:  warning\_btn\_yes = True self.warning\_type == self.WARNING\_TYPE.WARNINGSAVE | * Method update\_textfield\_input   is called once.   * Method execute\_reset is called once. * Method update\_textfield\_input * is called once with arguments: (True, False, False) |  | Pass |  |  |  |
| 13 | update\_warning\_btn\_yes  setup:  warning\_btn\_yes = True self.warning\_type = self.WARNING\_TYPE.RESET | * Method execute\_reset is called once. * Method update\_textfield\_input * is called once with arguments: (True, False, True) |  | Pass |  |  |  |
| 14 | update\_warning\_btn\_yes  setup:  warning\_btn\_yes = False self.warning\_type = self.WARNING\_TYPE.RESET | * Method update\_textfield\_input is called once with arguments: (False, True, False) |  | Pass |  |  |  |
| 15 | update\_widgets\_status  setup: reset\_btn\_disabled = False textfield\_disabled = True image\_saver\_dismiss = True | * Variable value reset\_btn\_disabled matches * Variable value textfield\_disabled matches * Variable value image\_saver\_dismissmatches |  | Pass |  |  |  |
| 16 | update\_main\_widgets  setup:  main\_text\_field.text = ''  display\_image.max\_available\_chars = 100 | * Variable value main\_image.source matches * Variable value textfield\_str matches * Variable value maximum\_char\_count matches * Variable value encodable\_bool matches |  | Pass |  |  |  |
| 17 | update\_main\_widgets  setup:  display\_image.decode\_image returns “decoded\_msg”  display\_image.max\_available\_chars = 11 | * Variable value main\_image.source matches * Variable value textfield\_str matches * Variable value maximum\_char\_count matches * Variable value user\_notification\_msg matches * Variable value encodable\_bool matches |  | Pass |  |  |  |
| 18 | update\_main\_widgets  setup:  display\_image.decode\_image returns “decoded\_msg”  display\_image.max\_available\_chars = 10 | * Variable value main\_image.source matches * Variable value textfield\_str matches * Variable value maximum\_char\_count matches * Variable value user\_notification\_msg matches * Variable value encodable\_bool matches |  | Pass |  |  |  |
| 19 | update\_textfield\_input  setup:  textfield\_str ="somestring" | * Variable value main\_text\_field.text matches |  | Pass |  |  |  |

**Table 4**

Manual Tests

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Test Input | Expected | Pass/Fail | | | | Fig. # |
| S1 | S2 | S3 | S4 |
| 1 | Randomly selected image loads on launch | Image displays |  | Pass |  |  |  |
| 2 | Image is decoded when loaded | Decoded message displays |  |  |  |  |  |
| 3 | Open image button opens file chooser | File chooser displays |  |  |  |  |  |
| 4 | File chooser has read access to user file system | Able to navigate file system |  |  |  |  |  |
| 5 | Attempt to open non-image file | File chooser rejects non-image files |  |  |  |  |  |
| 6 | Cancel button in the File chooser view is pressed | Return to the main GUI |  |  |  |  |  |
| 7 | Attempt to open image file | Image opens and is displayed |  |  |  |  |  |
| 8 | Attempt to enter test phrase into text input area | Text is displayed. Allowed character count decreases |  |  |  |  |  |
| 9 | Attempt to enter text greater than maximum allowed for the image | Warning message displays in the Main GUI. |  |  |  |  |  |
| 10 | Attempt to enter text greater than maximum allowed for the image, then encode image button is pressed. | Error dialog popups. Not encodable. |  |  |  |  |  |
| 11 | Enter valid characters in the textfield, then encode image button pressed | Text input is disabled  Reset button is enabled |  |  |  |  |  |
| 12 | Save image button pressed | Save file dialog opens |  |  |  |  |  |
| 13 | Attempt to save file with invalid file name “123 ” which end with a space. | Error dialog popups. |  |  |  |  |  |
| 14 | Attempt to save file with incorrect extension “123.JNG” | New image is saved with the file name “123.JNG”.  Successfully saved image dialog popups. |  |  |  |  |  |
| 15 | Save image button pressed. Select Cancel button inn save dialog | Returned to main screen |  |  |  |  |  |
| 16 | Overwrite existing filename | Warning pop up. Allows overwrite |  |  |  |  |  |
| 17 | Reset image after encoding. Then Yes button in the warning popup is pressed. | Warning pop ups. Image resets to original value |  |  |  |  |  |
| 18 | Reset image after encoding. No button in the warning popup is pressed. | Returned to main GUI. |  |  |  |  |  |
| 19 | Open image containing message to decode | Decoded message displays in text area. File name is displayed in the title bar. Image is displayed in the main window. |  |  |  |  |  |

# Results

The automated tests were run on systems S1 – S4. All tests passed on every system. Figure 1 shows the results of the automated tests run on S2.

Graphical user interface, text

Description automatically generated

*Figure 1*. System 2 automated unit test results.