Interactive Watermark Identification Procedure Project Report

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Hao Lu (hl935)

MEng Field Advisor: C. Richard Johnson, Jr.

Degree Date: May, 2016

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Interactive Watermark Identification Procedure

Author: Hao Lu (hl935)

Abstract: The aim of the project is to develop an interactive watermark identification procedure, which will reduce the workload and save time for art historians in identifying watermark types. For this purpose, we created an interactive decision tree for the identification procedure and implemented the decision tree as a website. The users can easily get their final results by answering the questions on the decision tree regarding certain features about their watermarks. Besides the website, we also did some literature research regarding the approaches to more fully automating the watermark identification procedure.

Executive Summary

The motivation behind the study of watermarks is to assist in the tracing of old documents and artifacts to provide plausible historical relationships and background information, such as date and origin. The problem is that there are so many different types of watermarks that need to be identified. Sometimes, it is really hard to tell the difference between two watermarks by just comparing them without knowing where to look. Also, the qualities of some of the beta radiographs are not so good. Sometimes it's hard to see the watermarks in the image. The poor image quality increases the difficulty for people to identify the watermark type manually. It requires a huge amount of time for a person to indentify just one watermark.

The motivation of this watermark identification project is to develop software to assist art historians with identifying watermark types, which will reduce the workload and save time for the art historians.

The watermark identification decision tree is implemented as a website. The website is composed of code written in HTML, CSS and JavaScript. On this website, users can upload the watermark images they have. The images will show up on the left side of the page. Users can easily compare the images they have with the sample images on the right side of the page. Users can also use the embedded image editor functions to make the watermark on the image easier to see. The website will guide users going through the decision tree by asking users some questions about the images they have. Users can easily follow the instructions and sample images to answer these questions. In the end, the final result about the classification of the watermark will show up.

Currently, only few of the branches on the decision tree has been created on the website. The future developers can use the template codes created to expand the decision tree easily. The attempt to automate the watermark identification procedure encountered some huge challenges. The biggest problem is the quality of the radiograph images. Different images may have different exposure levels. It makes the extraction of a clear watermark from the radiograph image really hard. Once the extraction problem can be resolved, it would be easier to compare the similarity of two watermarks.

Introduction

The motivation behind the study of watermarks is to assist in the tracing of old documents and artifacts to provide plausible historical relationships and background information, such as date and origin. The problem is that there are so many different types of watermarks that need to be identified. For example, in Rembrandt' prints, there exist categories like Foolscap, Basilisk, Amsterdam Arms, Eagle, Letters. According to Dr. Erik Hinterding's study on watermarks on Rembrandt's etchings, under the Foolscap category, there are 19 different variants. For each variant, there are several different sub-variants. For some sub-variants, they may also have twin watermarks. The twin watermarks are extremely similar to each other but they are not identical. As a result, for someone who is not as familiar with the watermarks as Erick Hinterding, that person needs to compare the watermark s/he has with over twenty other watermark examples. Sometimes, it is really hard to tell the difference between two watermarks by just comparing them without knowing where to look. Also, the qualities of some of the beta radiographs are not so good. Sometimes it's hard to see the watermarks in the image. The poor image quality increases the difficulty if identifying the watermark type manually. It can require a substantial amount of time for a person to indentify just one watermark.

The motivation of this watermark identification project is to develop software to assist art historians with identifying watermark types, which will reduce the workload and save time for the art historians. After the presentation of the software template for an interrogatory decision-tree, a brief review of methods from the literature regarding approaches to more fully automating the watermark identification process concludes this report.

Issues to Be Addressed.

There are several issues that need to be addressed in this project. First of all, the dataset the project team has is limited but not exhaustive and the initial decision tree the team created is not 100% accurate. The project team only has images from the collection

of the Dutch University Institute for Art History in Florence. To make the decision tree better and more accurate, the project team will eventually need to get more help from Dr. Hinterding.

The second issue for the project to address is to research and develop algorithms automating the identification procedure. If the project team can automate parts of the procedure, it will greatly reduce the workload for art historians.

Website Design:

Based on the dataset images the team already has and the study of Dr. Hinterding on classifying watermark types[1], students created a decision tree for identifying Fools Cap type of watermarks. The decision tree includes questions such as “Are roundels in a pyramid form? “,”Are the peaks divided (stripes in cap)?”,”Are there circles across the brow?”. Different answers to the questions will lead to a different path on the decision tree. At the end of the path, the decision tree will indicate the classification of the watermark.

Engineering students developed software based on the decision tree provided by the other students in the team. The project team decided to create a demo version first to show Dr. Hinterding and get his advice on this project. In the demo version, only one branch of the decision tree was developed. The engineering students used html, css, and javascript to create pages for each step along the path and link them based on the decision made on the previous page. In the demo version, the user can upload the image s/he wants to identify and edit the images. For example, the user can change the brightness, contrast, and saturation of the image to make the watermark on the image clearer. The user can also flip the image horizontally and vertically to make the orientation of the watermark the same as the stored samples in order to make to comparison easier. In each page, there is text instruction and a sample image telling the user where to look and compare. The differences are marked in red on the example images in order to give users a better understanding.

User Guide

The users need to first download the watermark v2 folder onto their computer in order to use the demo version watermark identification program[1]. After downloading the folder, users can start using the program by clicking on the index.html file under the v2 folder. Fig. 1 shows the start screen.

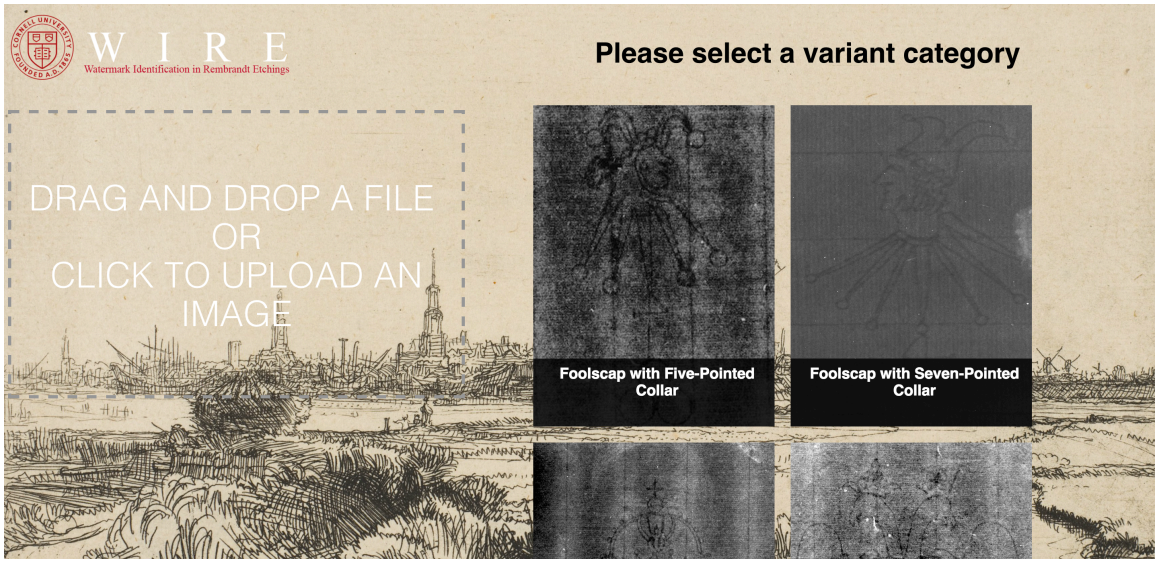

Fig.1 The user interface of the software start screen

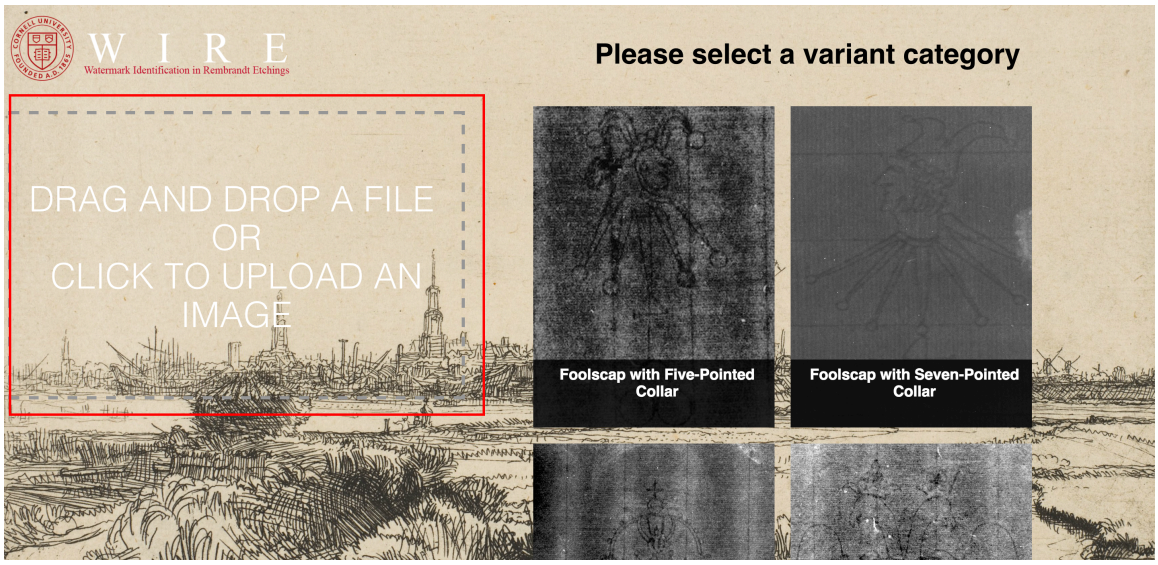

Fig.2 The area where users can upload images is marked in red

[1] The software V2 folder is available from Andrew Weislogel, a curator at the Herbert F. Johnson Museum of Art at Cornell University, who will lead future teams in expanding this effort

As shown in Fig.2, the area marked in red on the left of the screen is used for uploading images. Users can upload the image they want to test on to the program by dragging the image file to this area or clicking on the area to choose image file from the computer.

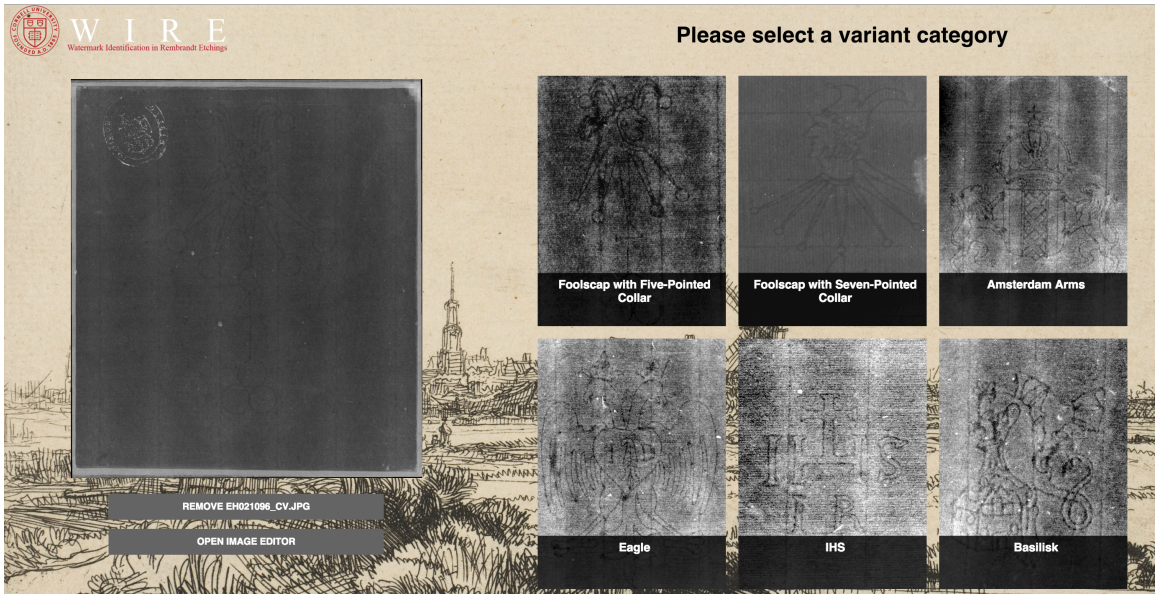

Fig.3 The user interface after uploading images

After uploading the image to the program, two buttons will appear below the uploaded image as shown in Fig.3. One is used to remove the image from the program so that users can choose another image to upload. The other one is used to open image editor, which the users can use to adjust the images as they want. Then the users can choose the category of their watermark being identified by clicking on the options on the right of screen as shown in Fig. 4.

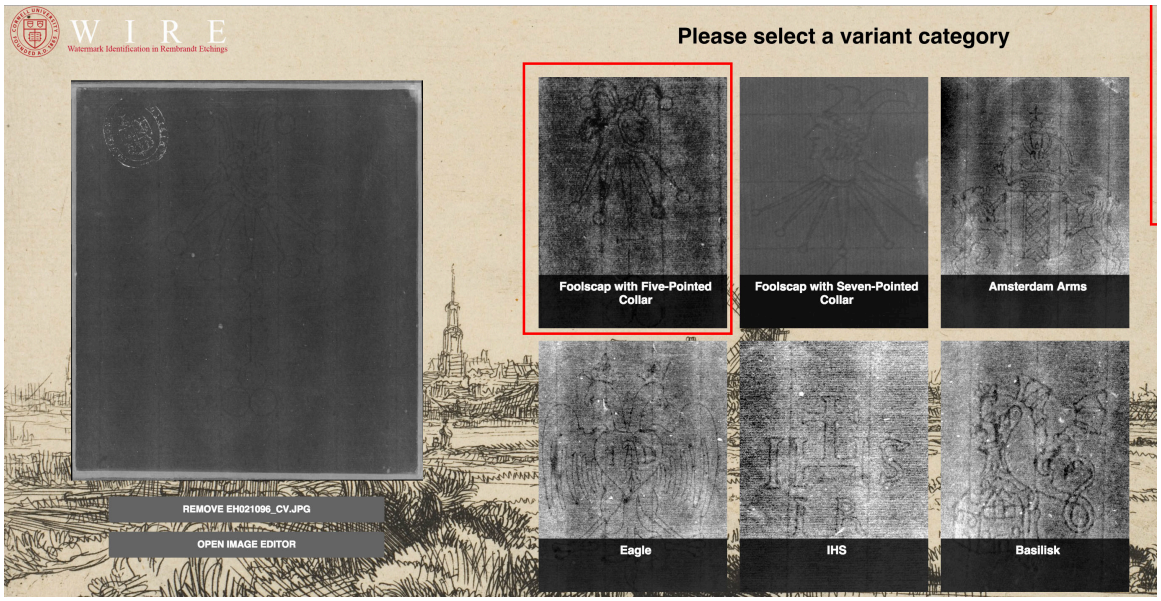

Fig.4 The area marked in red shows one category that can be identified

After choosing the category, users will need to answer some questions about their watermark. The questions will show up on the top right of the screen as in Fig.5.

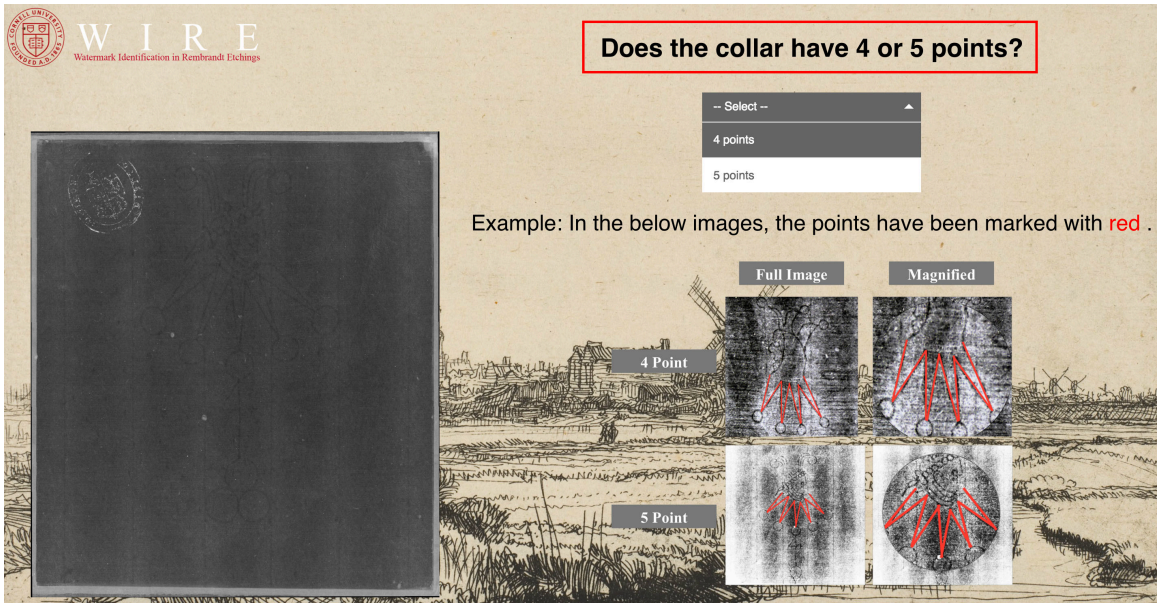

Fig.5 The area marked in red is used for displaying questions

The instructions and example images will show up on the bottom right of the screen (see Fig. 6) to give users a better understanding about the feature they should examine and compare.

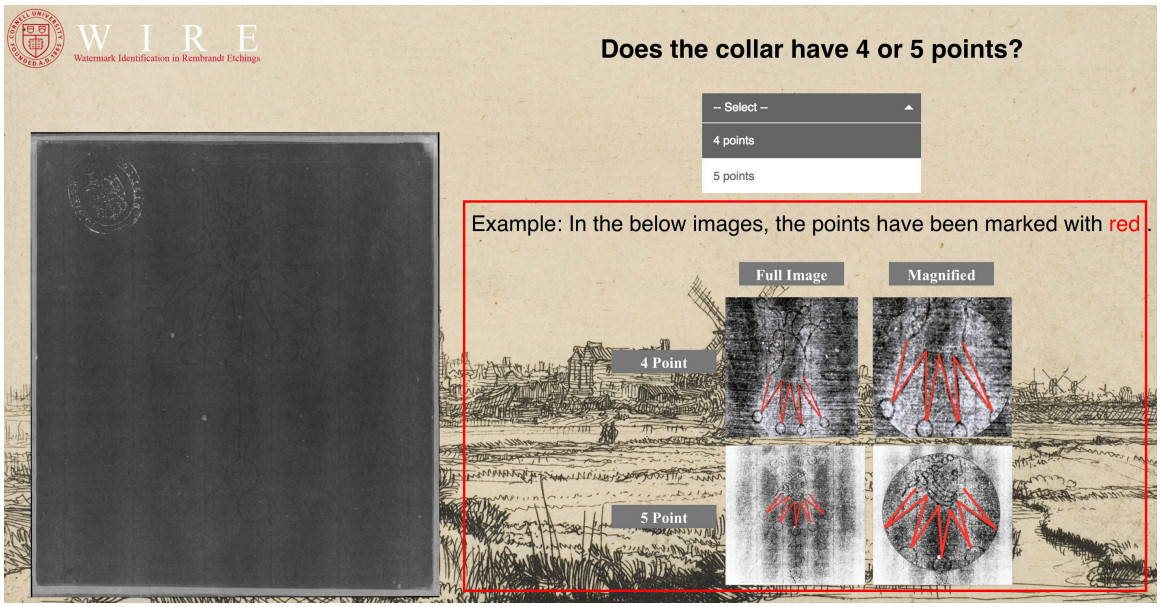

Fig.6 The area marked in red is used for displaying detailed instructions and sample images

Users can choose their answers from the option boxes as in Fig.7.

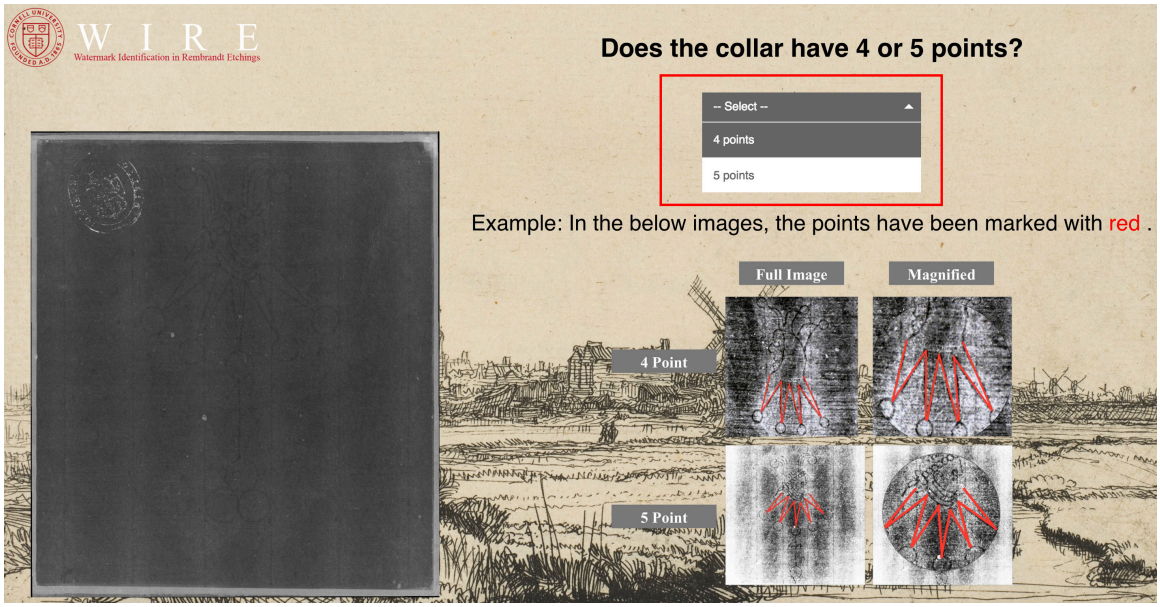

Fig.7 The area marked in red is used for option boxes

Then users can click on the confirm button to submit the answer and the website will lead users to a new page.

After following the same procedure and answering several more questions, users will get a final result from the website as shown in Fig.8.

Fig.8 Final result page sample

The instructions for people who have no programming experience:

Fig.9 displays an example file directory system.

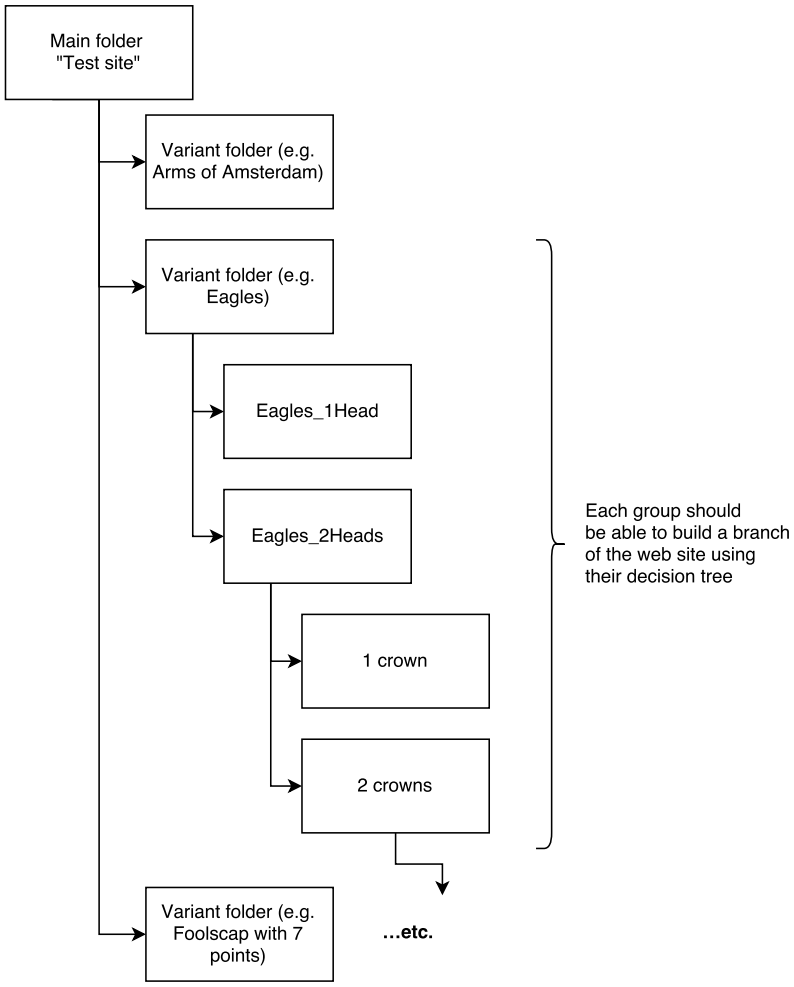

Fig.9 File directory system used to develop the software

List of things needed to add a branch:
- text editor
- images of watermark in each branch
- edited image details that illustrate the options in your branch

1. Install a text editor for modifying HTML files.

The developers first need to download a file editor such as Brackets in order to open and modify the files used in the website. The program can be downloaded at http://brackets.io/. After installing Brackets, developers can start building new branches based on the templates presented here.

STEPS (Mac)
- Download Brackets.dmg and open it
- Drag the Brackets application to your applications folder

A web browser is needed to view your progress.

2. Create a new folder for the new branch.

In the current design of the website (March 2016), in order to make a new decision tree branch, developers will need to create a new folder for that branch. This

new folder will include a CSS file folder, a JS file folder, an index.html file, a background image bg.png and a sample image file used for instructions.
STEPS
- In your file browser (e.g. Finder on Mac), navigate to the Test File folder.
- Create a new folder for your branch of general watermark type (e.g. Eagles, ArmsOfAmsterdam). Name it in the following form: "Type_Step1" (e.g. Eagles_Step1). Avoid spaces- use underscores.

3. Fill the new folder with items in the sample folder

After creating the new folder, copy and paste everything in the sample except (kba folder, kbb folder, step8.png and step8 copy.png) into the new folder. The items that don't need to be copied have been crossed out as shown in Fig. 10.

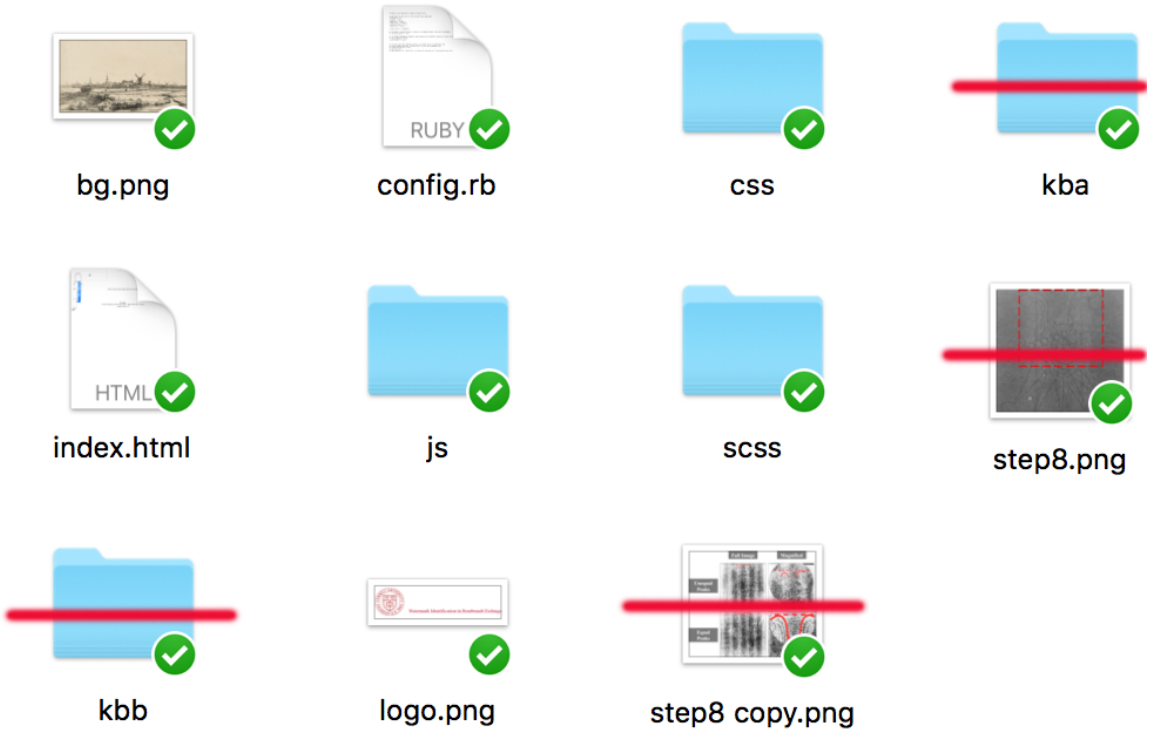

Fig. 10 The files used for new branch development.(The files that don't need to be copied are crossed out.)

4. Set up folders for the next steps in the branch.

Create empty folders for the possible answers to the next steps in your branch. For example, from the main Eagles category, the next steps are "1 head" or "2 heads." Name the folders for the next steps, using the variant as the main name and adding an underscore followed by a short comment or text to this file name to indicate what the step is about (e.g. Eagles_1Head, Eagles_2Heads, etc.)

5. Change text in files in the JS file folder

In the CSS file folder, there are three files: editor.css, upload.css and main.css. Normally there is nothing in these files that needs to be changed. In the JS file folder, there are also three files: editor.js, upload.js and main.js. Developers only need to modify

the final link function in the main.js file as shown in Fig. 11 when they create a new branch.

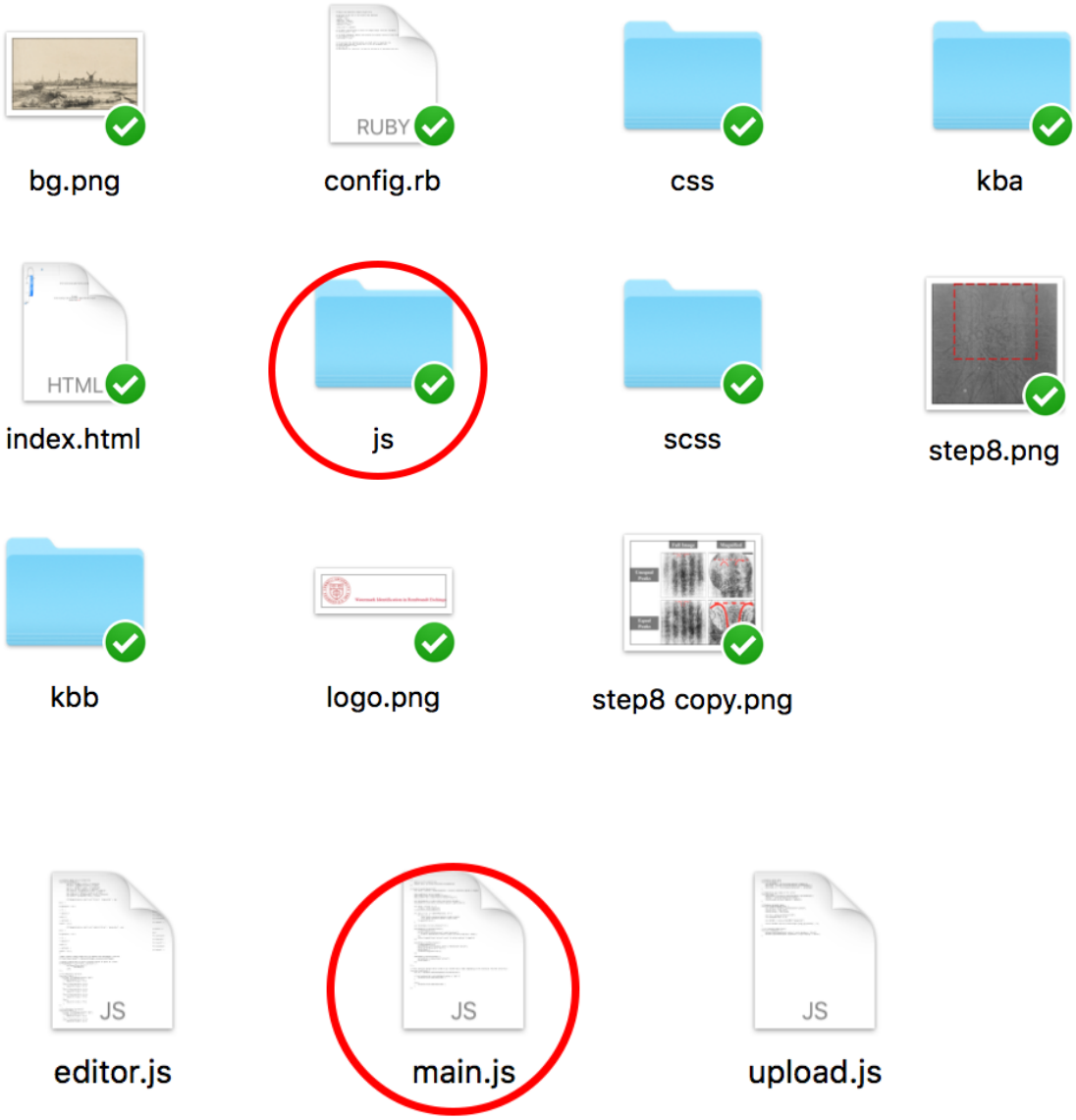

Fig. 11 The main.js file that needs to be modified can be found in the js folder

Developers only need to modify the text in orange after the equal sign as shown in Fig. 12. They should change the text to the directory of the page to which they want the option linked. In our example, the next step folders are called Kba and Kbb. We want to link the current webpage to the index.html file in the next step folder. As shown in the example, the directory is ***kba/index.html*** and ***kbb/index.html***. In the example, the first option will link the current webpage to the index.html page inside the kba folder and the second option will link the page to the index.html page inside the kbb folder. The developers should change the orange text before the '/' sign to the folder name they have created.

STEPS:

- Open ***main.js*** in Brackets or a similar text editor.
- Find the section called function ***finalLink***, around ***line 53***
- Change the text in the "if" section ("kba" in example) to the name of new branch folder options. There are two places to do this, value == 'NEW_NAME' and location.href='NEW_NAME/index.html'
- Change the text in the "else" section ("kbb in example) to the name of the other branch folder option
- If there is a 3rd option, put an else if statement between } and else, the format should be: ***else if (sel.option[sel.selectedIndex].value=='NEW_NAME2') { location.href='NEW_NAME2/index.html';}***

```
// This function defines which index to go (either kba or kbb) depending on the selection from the
select box
function finalLink() {
    var sel = document.getElementById('finalSelection');

    if (sel.options[sel.selectedIndex].value == 'kba') {
        location.href='kba/index.html';
    }

    else {
        location.href='kbb/index.html';
    }
};
```

Fig.12 The final link function that needs to be modified

6. Edit the index.html file.

Open the index.html in Brackets. In the index,html file, there are several parts require modification. The first one is the show main question part as shown in Fig.13. This section can be easily found around ***line 110***.

Developers can change the text in black to the question they want to ask.

```
<!-- Show main question -->
<div class="mainQuestion">
    <p align="center" style="color:black;">Is the foremost peak higher than the rear peak?
    </p>
</div>
```

Fig.13 The codes related to the questions that need to be modified and the corresponding area on the actual page

The second one is the select option part as shown in Fig.14. This section can be found around ***line 115***.

Developers can change "Yes" and "No" to any other answers they want to use. In order put additional options, developers should follow this format: <option value= "xxx"> "new option" </option> and add this before </select>.

```
<!-- Select option -->
<div class="optionWrapper">
    <select id="finalSelection">
        <option value="hide">-- Select --</option>
        <option value="kba" rel="icon-temperature">Yes</option>
        <option value="kbb">No</option>
    </select>
</div>
<br/><br/>
```

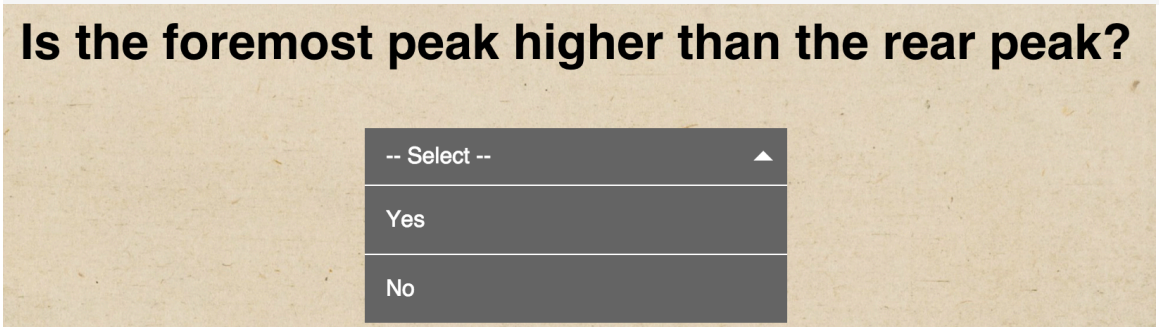

Fig.14 The codes related to options that need to be modified and the corresponding area on the actual page

For clarification, the option value can be the name of the new branch the answer will lead to. Remember, the option value here must be consistent with the option value used in the ***finalLink function***.

The last one need to be modified is the show instruction image part as shown in Fig. 15. This part can be found around ***line 132***.

```
<!-- Show instruction image -->
<div class="instruction">
    <p align="center">Example:</br>In the top images, the foremost peak is higher than the
    rear peak</br/>(marked with <span style="color: red;">red</span>).</p>
    <img src="step8 copy.png">
</div>
```

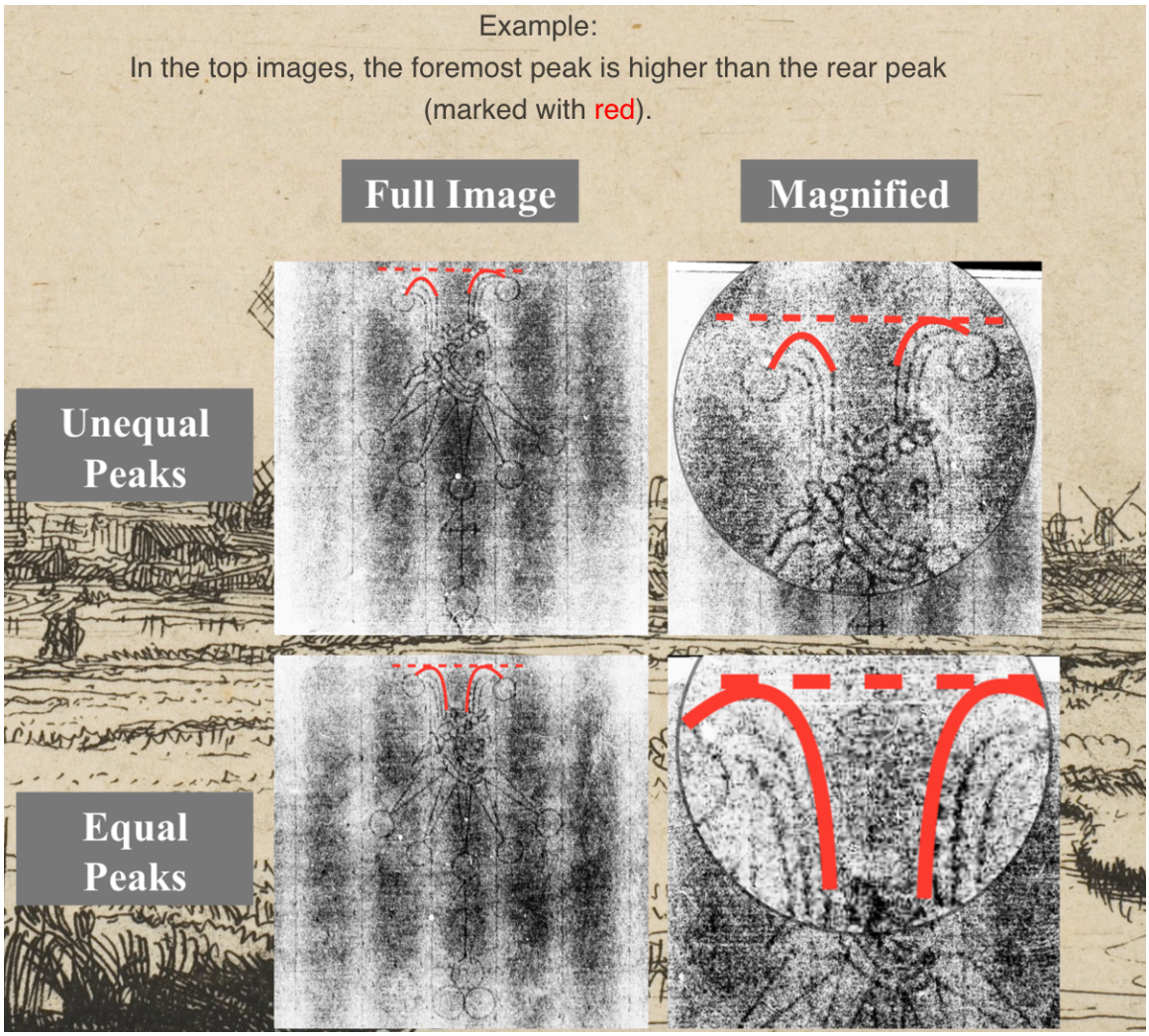

Fig. 15 The codes related to the instructions and sample images and the corresponding area on the actual page.

The black text between </br> and </br> can be changed to other instructions that are suitable for the decision tree branch. Meanwhile, the orange text "step8 copy.png" should be changed to the new example image that is suitable for this branch.

Automated Classification Algorithm Research:

This section summarizes a few approaches to automation of watermark identification found in the open literature.

Literature Research:

In the thesis 'Content-based Paper Retrieval Towards Reconstruction of Art History' [2]written by Dr. Mark Van Staalduinen, one approach in attempt to find paper matching is to use the features of laid and chain lines. The approach suggests that two

papers, which have similar chain line spaces, are possibly matches. The paper matches should have the same watermarks.

However, there are still some issues with such approach. It is possible for two papers, which have different watermarks to have similar chain line spaces. It means the accuracy of such approach may not as desired.

In the paper 'Retrieval of Images from a Library of watermarks for ancient Paper Identification' [3] written by Christian Rauber, Peter Tschudin and Thierry Pun, there are two algorithms presented in [3] for comparing similar images. The first one is called circular histogram algorithm. The idea is to divide the watermarks equally into 16 quadrants and compute a circular histogram around the center of gravity of the watermark. Ideally two watermarks can be considered globally similar if their respective histograms are similar. The resemblance d(H1, H2) between two histograms is computed as following:

d(H1, H2)=0;
for i=1:16
d(H1,H2)=d(H1, H2)+| H1[i]-H2[i] |;
end

The second method is called directional algorithm. The second algorithm consists of filtering the input image by height directional filters:

$Gj(x, y)=I(x, y)*Fj \quad j=1\ldots.8$

After this operation, eight new planes are obtained. Then take the highest value from these planes to compute K(x, y).

$Kj(x, y)=Gj(x,y)$ if $Gj(x,y)=\max(Gj(x, y))>1$

$Kj(x, y)=0$ otherwise

The sum of K can be compared in order to find similar watermarks.

There are some challenges to these two approaches. First of all, the quality of the original images really matters. In some images, currently the grey scale of the watermark is really close to the background. The difference in the watermark grey scale level will significantly affect the histograms of the images. As a result, the poor quality of the image may lead to a false result.

In the paper ‘A System for Segmenting and Extracting Paper-based Watermark Designs’ [4]written by Hazem Hiary and Kia Ng, an approach to enhance the clarity of the watermark image is presented. They suggest that a Canny detector can be used to locate edges of the watermark. Then a noise removal process is applied. Small gaps between image features can be eliminated by applying a morphological closing operation.

There are still some challenges to such an approach. In order to get a desirable enhancement, the noisy image features have to be small and isolated. It also means that the quality of the original image really matters. Also, different images need to have different setting parameters to enhance the quality. It would require a significant amount of a work to get the dataset currently available properly enhanced.

Other Ideas

Comparing the similarity of two whole images currently may not be an effective and efficient approach. Instead, the whole procedure can be divided into smaller segments. In each segment, we only need to care about certain features in that segment. In this way, the calculation and comparison would be easier for computer. For example, at the end of Kbb and Kba branch, the only thing we need to do is to decide if one peak is higher than the other. At this stage, we assume that the computer would know exactly where it should focus. We could try to extract a rectangular area. The upper bound should be at the same height as the higher peak. The height of this rectangular area should be equal to the difference between the heights of higher peak and lower peak. The length of the area should be just enough to cover the watermark from left to right. Then we could divide the area into halves and compute the histogram bin counts for the left half and for the right half. Ideally, if the difference of bin counts is large, possibly one peak is higher than the other since one part contains some part of the peak while the other part doesn’t have any watermark part inside. Otherwise, if the bin counts are close, possibly the two peaks have the same height.

After making these segments automated, we could possibly combine them together in order to make the whole decision tree fully automated.

However, there are still some issues need to be addressed before such approach can be implemented. The first one is to find an effective way to extract a clear watermark image out. If we cannot get a clear watermark, it would be really difficult to do the

calculation and comparison since the results may not be accurate. The second one is to find an effective way to finish segmentation. We need to let the computer focus on a certain area in order to reduce the calculation and improve the accuracy.

If we can get these issues solved, the progress of making the whole decision tree automated will make a leap forward.

References

[1] Erik Hinterding, Rembrandt as an Etcher: The Practice of Production and Distribution, (dissertatie) Ouderkerk aan den IJssel, 3 vols., 2006.

[2] M. van Staalduinen, "Content-based Paper Retrieval Towards Reconstruction of Art History," PhD Thesis, TU Delft, 2010.

[3] Rauber, C., Tschudin, P., and Pun, T. (1997b), Retrieval of images from a library of watermarks for ancient paper identification, EVA'97 - Elektronische Bildverarbeitung und Kunst, Kultur, Historie, November 12-14, Vol. 14.

[4] Hazem Hiary, Kia Ng ,A System for Segmenting and Extracting Paper-based Watermark Designs *International Journal on Digital Libraries*, Vol. 6, No. 4. (2007), pp. 351-361 Hazem Hiary, Kia Ng

Appendix A

Commented Code Listing

There are three types of files are used to create a webpage including html, css and javascript. Html file is used to edit the structure and texts of the page. Css file is used to format and decorate the page in order to make it look nicer. Javascript is used to create and add some functions to the page.

1. Instructions about the html files

index.html for the front page

```
<!doctype html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Watermark Project</title>

   <!-- Upload Image-->
        <link rel="stylesheet" href="css/upload.css">
        <script src="js/upload.js"></script>
   <script class="jsbin"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script
>

   <!-- Main -->
        <link rel="stylesheet" href="css/main.css">

   <!-- Editor -->
   <link href='http://fonts.googleapis.com/css?family=Source+Sans+Pro'
rel='stylesheet' type='text/css'>
```

```
    <link href="http://fonts.googleapis.com/css?family=Inconsolata"
rel="stylesheet" type="text/css">
    <link rel="stylesheet" href="http://materializecss.com/css/ghpages-
materialize.css">
    <link rel="stylesheet" href="css/editor.css">

</head>

<body>
    <!-- Left Side -->
    <div class="left-side">
        <!-- Logo -->
            The developers can change the logo image by changing the source
of the image after ‘src=’. The developers can also define the size of the
logo after ‘style=’.
        <img src="logo.png" style="width:350px; height:100px;">

        <!-- Upload Image-->
        <div class="file-upload">
            <div class="image-upload-wrap">
                <input class="file-upload-input" type='file'
onchange="readURL(this);" accept="image/*" />
                <div class="drag-text">
The developers can change the text message in the upload image block by
editing the text between <h3> and </h3>
                    <h3>Drag and drop a file </br>or</br/>click to upload an
image</h3>
                </div>
            </div>

            <div class="file-upload-content">
```

```
            <img class="file-upload-image" id="uploadedImage" src="#"
alt="your image" />
            <img class="copy-upload-image" id="copyImage"
style="display:none;"/>

            <div class="image-title-wrap">
```
The code below defines the remove image button. ‘removeUpload()’ is the function that will be called when users click on the button.
```
              <button type="button" onclick="removeUpload()"
class="remove-image">Remove <span class="image-title">Uploaded
Image</span>
              </button>
            </div>
          </div>

          <div class="file-upload-content">
            <div class="image-title-wrap">
```
The code below defines the open image editor button. ‘showEditor()’ is the function that will display the image editor when users click on the button.
```
              <button type="button" onclick="showEditor();"
class="image-editor">Open Image Editor</span>
              </button>
            </div>
          </div>
        </div>

        <!-- Flip Button -->
        <div id="flipButton" style="display:none;">
          <div class="file-upload-content">
            <div class="image-title-wrap">
```

The code below defines the flip image horizontally button. The function ‘fliph()’ is called when users click on the button. The button is named as ‘Flip Horizontal’

```
            <button type="button" onclick="fliph();" class="rotate">Flip
Horizontal
            </button>
            </div>
          </div>

          <div class="file-upload-content">
            <div class="image-title-wrap">
```

The code below defines the flip image vertically button. The function ‘flipv()’ is called when users click on the button. The button is named as ‘Flip Vertical’

```
            <button type="button" onclick="flipv();" class="rotate">Flip
Vertical
            </button>
            </div>
          </div>
        </div>

        <!-- Editor -->
        <div id="editor" style="display:none;" class="container">
          <div class="row">
            <form id="imageEditor">
```

The code below defines the value ranges of sliders for each option under the image editor.

```
              <div class="sliders col s4">
                <p class="range-field">
                  <label for="gs">Grayscale</label>
```

```
                    <input type="range" id="gs" name="gs" min="0"
max="100" value="0"/>
                </p>
                <p class="range-field">
                    <label for="blur">Blur</label>
                    <input type="range" id="blur" name="blur" min="0"
max="10" value="0"/>
                </p>
                <p class="range-field">
                    <label for="br">Exposure</label>
                    <input id="br" name="br" type="range" min=0 max=200
value=100>
                </p>
                <p class="range-field">
                    <label for="ct">Contrast</label>
                    <input id="ct" name="ct" type="range" min=0 max=200
value=100>
                </p>
            </div>

            <div class="sliders col s4">
                <p class="range-field">
                    <label for="opacity">Opacity</label>
                    <input id="opacity" name="opacity" type="range"
min=0 max=100 value=100>
                </p>
                <p class="range-field">
                    <label for="invert">Invert</label>
                    <input id="invert" name="invert" type="range" min=0
max=100 value=0>
                </p>
```

```
                    <p class="range-field">
                        <label for="saturate">Saturate</label>
                        <input id="saturate" name="saturate" type="range"
min=0 max=500 value=100>
                    </p>
                    <p class="range-field">
                        <label for="sepia">Sepia</label>
                        <input id="sepia" name="sepia" type="range" min=0
max=100 value=0>
                    </p>
                </div>
                <br/><br/>
```

The code below defines the reset image editor button.

```
                <button class="btn" type="reset" id="reset">Reset</button>
            </form>
        </div>
    </div>
</div>

<!-- Main Page -->
<div id="mainwrapper">
    <div class="mainQuestion">
```

The code below defines the position and color of the instruction text. The developers can change the text between and

```
        <p align="center" style="color:black;">Please select a variant
category</p>
    </div>

    <div id="boxWrapper">
```

The developers can change the page the image block will link to by altering the url after ‘href=’.

```
        <div id="box-1" class="box"
onclick="save();location.href='step2/index.html';">
            <div id="bg-block" class="bg-block-1"></div>
            <span class="caption simple-caption cap1">
```

The code below defines the text in the image block.

```
                <div class="title">Foolscap with Five-Pointed Collar</div>
            </span>
        </div>

        <div id="box-1" class="box">
            <div id="bg-block" class="bg-block-2"></div>
            <span class="caption simple-caption cap2">
```

The code below defines the text in the image block.

```
                <div class="title">Foolscap with Seven-Pointed Collar</div>
            </span>
        </div>

        <div id="box-1" class="box">
            <div id="bg-block" class="bg-block-3"></div>
            <span class="caption simple-caption cap3">
```

The code below defines the text in the image block.

```
                <div class="title">Amsterdam Arms</div>
            </span>
        </div>

        <div id="box-1" class="box">
            <div id="bg-block" class="bg-block-4"></div>
            <span class="caption simple-caption cap4">
```

The code below defines the text in the image block

```
                <div class="title">Eagle</div>
            </span>
        </div>

        <div id="box-1" class="box">
            <div id="bg-block" class="bg-block-5"></div>
            <span class="caption simple-caption cap5">
The code below defines the text in the image block
                <div class="title">IHS</div>
            </span>
        </div>

        <div id="box-1" class="box">
            <div id="bg-block" class="bg-block-6"></div>
            <span class="caption simple-caption cap6">
The code below defines the text in the image block
                <div class="title">Basilisk</div>
            </span>
        </div>
    </div>
        </div>

    <!-- Editor -->
    <script
src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></scr
ipt>
    <script src='http://materializecss.com/bin/materialize.js'></script>
    <script src="js/editor.js"></script>
</body>
</html>
```

index html for the other steps

Most of the code is the same as the index.html for the front page. The parts that are different will be explained in red.

```
<!doctype html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Watermark Project</title>

   <!-- Upload Image-->
        <link rel="stylesheet" href="css/upload.css">
        <script src="js/upload.js"></script>
   <script class="jsbin"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script
>

   <!-- Main -->
        <link rel="stylesheet" href="css/main.css">

   <!-- Editor -->
   <link href='http://fonts.googleapis.com/css?family=Source+Sans+Pro'
rel='stylesheet' type='text/css'>
   <link href="http://fonts.googleapis.com/css?family=Inconsolata"
rel="stylesheet" type="text/css">
   <link rel="stylesheet" href="http://materializecss.com/css/ghpages-
materialize.css">
   <link rel="stylesheet" href="css/editor.css">
```

Below is the javascript code that load the image uploaded in the front page to the new page.

```
    <script>
    function getImage() {
        var dataImage = localStorage.getItem('imgData');
        bannerImg = document.getElementById('uploadedImage');
        bannerImg.src = "" + dataImage;
    }
    </script>
</head>

<body onLoad="getImage();">
    <!-- Left Side -->
    <div class="left-side">
        <!-- Logo -->
        <img src="logo.png" style="width:350px; height:100px;">

        <!-- Show Image -->
        <div id="imageContainer" >
            <img src="" id="uploadedImage" class="center" style="margin-
top: 50px;"/>
            <img class="copy-upload-image" id="copyImage"
style="display:none;"/>
        </div>

        <button type="button" onclick="showEditor();" class="image-
editor">Open Image Editor</span></button>
    <br/>

        <!-- Flip Button -->
        <div id="flipButton" style="display:none;">
            <div class="file-upload-content">
                <div class="image-title-wrap">
```

```
            <button type="button" onclick="fliph();" class="rotate">Flip
Horizontal
            </button>
            </div>
          </div>
          <br/>
          <div class="file-upload-content">
            <div class="image-title-wrap">
            <button type="button" onclick="flipv();" class="rotate">Flip
Vertical
            </button>
            </div>
          </div>
       </div>
    <br/>

       <!-- Editor -->
       <div id="editor" style="display:none;" class="container">
          <div class="row">
             <form id="imageEditor">
                <div class="sliders col s4">
                   <p class="range-field">
                      <label for="gs">Grayscale</label>
                      <input type="range" id="gs" name="gs" min="0"
max="100" value="0"/>
                   </p>
                   <p class="range-field">
                      <label for="blur">Blur</label>
                      <input type="range" id="blur" name="blur" min="0"
max="10" value="0"/>
                   </p>
```

```
                <p class="range-field">
                  <label for="br">Exposure</label>
                  <input id="br" name="br" type="range" min=0 max=200
value=100>
                </p>
                <p class="range-field">
                  <label for="ct">Contrast</label>
                  <input id="ct" name="ct" type="range" min=0 max=200
value=100>
                </p>
              </div>

              <div class="sliders col s4">
                <p class="range-field">
                  <label for="opacity">Opacity</label>
                  <input id="opacity" name="opacity" type="range"
min=0 max=100 value=100>
                </p>
                <p class="range-field">
                  <label for="invert">Invert</label>
                  <input id="invert" name="invert" type="range" min=0
max=100 value=0>
                </p>
                <p class="range-field">
                  <label for="saturate">Saturate</label>
                  <input id="saturate" name="saturate" type="range"
min=0 max=500 value=100>
                </p>
                <p class="range-field">
                  <label for="sepia">Sepia</label>
```

```
                    <input id="sepia" name="sepia" type="range" min=0
max=100 value=0>
                    </p>
                </div>
                <br/><br/>

                <button class="btn" type="reset" id="reset">Reset</button>
            </form>
        </div>
    </div>
</div>

<!-- Main Page -->
<div class="main">
    <div class="mainQuestion">
```
The code below defines the position and color of questions. The developers can change questions by editing the content between and
```
        <p align="center" style="color:black;">Does the collar have 4 or 5
points?</p>
    </div>
```
The code below defines the option box.
```
    <div class="optionWrapper">
        <select id="mounth">
            <option value="hide">-- Select --</option>
            <option value="january" rel="icon-temperature">4
points</option>
            <option value="february">5 points</option>
        </select>
    </div>
```

```
        <br/><br/>
```
The code below defines the confirm button.
```
        <div id="confirm">
            <button type="button"
onclick="location.href='step3/index.html';">Confirm</button>
        </div>
```

The developers can define the color, position and content of the instruction below.
```
        <div class="instruction">
            <p align="center">Example: In the below images, the points have
been marked with <span style="color: red;">red</span> .</p>
```
The example image can be altered by changing the content after ‘src=’.
```
            <img src="step2.png">
        </div>
    </div>

    <!-- Editor -->
    <script
src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></scr
ipt>
    <script src='http://materializecss.com/bin/materialize.js'></script>
    <script src="js/editor.js"></script>

    <!-- Main -->
    <script
src='//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>
    <script src="js/main.js"></script>
</body>
</html>
```

2. Instructions about the css files

editors.css

The code below determines the appearance of the image editor block.

```
/*custom materialize*/
.container {
    display: block;
    margin-left: 22%;
}

label {
  display: inline-block;
  width: 200px;
  font-size: 17px;
  color: #fff;
}

input[type=range] {
  width: 100%;
  margin-left: 10px;
}

.col {
    display: block;
    margin-left: auto;
    margin-right: auto;
    background-color: #666666;
}

.btn {
  margin-top: 15px;
  width: 66.5%;
```

```
  background: #666666;
}

.btn:hover {
  background: #666666;
}

.btn:active {
  background: #666666;
}

.btn:focus {
  background: #666666;
}

input {
    background: transparent;
    outline: 0;
    border: 0;
    color: white;
}
```

f1,f2, f3, f4 are four possible status of image after flips.

```
.f1 {
    transform:scale(1,1);
    -ms-transform:scale(1,1)
    -webkit-transform: scale(1, 1);
    max-height: 700px;
    max-width: 600px;
}

.f2 {
```

```
    transform:scale(-1,1);
    -ms-transform:scale(-1,1)
    -webkit-transform: scale(-1, 1);
    max-height: 700px;
    max-width: 600px;
}

.f3 {
    transform:scale(-1,-1);
    -ms-transform:scale(-1,-1)
    -webkit-transform: scale(-1, -1);
    max-height: 700px;
    max-width: 600px;
}

.f4 {
    transform:scale(1,-1);
    -ms-transform:scale(1,-1)
    -webkit-transform: scale(1, -1);
    max-height: 700px;
    max-width: 600px;
}
```

main.css

```
@import url(http://fonts.googleapis.com/css?family=Roboto:300);

.mainQuestion p {
  color: #ffffff;
  font-size: 35px;
  font-weight: bold;
}
```

```
/* windows 8 tiles */
#mainwrapper {
  position: relative;
  background-color: transparent;
  width: 60%;
  float: right;
  overflow: auto;
}

#mainwrapper #boxWrapper {
    position: relative;
    display: block;
    margin-left: 75px;
}

#mainwrapper #tools{
  text-align:center;
  font-size:25px;
  margin-bottom:10px;
}

#mainwrapper .box .title{
        text-align:center;
        padding:8px 0px 0px 8px;
        font-weight:bold;
}
#mainwrapper .box p{
        margin: 3px 0px 0px 0px;
        padding:0px 0px 8px 8px;
        text-align:left;
}
```

```
#mainwrapper .box .simple-caption p{
  display:none;
}

#mainwrapper {
        font: 10pt normal Arial, sans-serif;
        height: auto;
        text-align: center;
        width: 60^px;/* Size of wrapper */
}

/* Image Box Style */
#mainwrapper .box {
   display: block;
        cursor: pointer;
        width: 300px;
   height: 400px;
        float: left;
        margin: 10px;
        position: relative;
        overflow: hidden;
}

#mainwrapper .box #bg-block{
  position: absolute;
  background-repeat:no-repeat;
  background-size:100% 100%;
  background-position:0px 0px;
  width:100%;
  height:100%;
  left: 0;
```

```
  -webkit-transition: all 300ms ease-out;
  -moz-transition: all 300ms ease-out;
  -o-transition: all 300ms ease-out;
  -ms-transition: all 300ms ease-out;
  transition: all 300ms ease-out;
}

/* Caption Common Style */
#mainwrapper .box .caption {
        background-color: rgba(0,0,0,0.8);
        position: absolute;
        color: #fff;
        z-index: 100;
    -webkit-transition: all 300ms ease-out;
        -moz-transition: all 300ms ease-out;
        -o-transition: all 300ms ease-out;
        -ms-transition: all 300ms ease-out;
        transition: all 300ms ease-out;
        left: 0;
}

/** Caption 1: Simple **/
#mainwrapper .box .simple-caption {
        height: 182px;
        width: 300px;
        display: block;
        bottom: -182px;
        text-align: center;

    -moz-transform: translateY(-30%);
        -o-transform: translateY(-30%);
```

```
        -webkit-transform: translateY(-30%);
        opacity: 1;
        transform: translateY(-30%);
}
#mainwrapper .box:hover .simple-caption p{
  display:block;
}
/** Simple Caption :hover Behaviour **/
#mainwrapper .box:hover .simple-caption {
        -moz-transform: translateY(-83%);
        -o-transform: translateY(-83%);
        -webkit-transform: translateY(-83%);
        opacity: 1;
        transform: translateY(-83%);
}
#mainwrapper .box:hover .scale-caption.title, #mainwrapper .box:hover
.scale-caption p {
        -moz-transform: translateX(200px);
        -o-transform: translateX(200px);
        -webkit-transform: translateX(200px);
        transform: translateX(200px);
}

#mainwrapper .box .cap1{
    background-color: ;
}
#mainwrapper .box .cap2{
    background-color: ;
}
#mainwrapper .box .cap3{
    background-color: ;
```

```
}
#mainwrapper .box .cap4{
    background-color: ;
}
#mainwrapper .box .cap5{
    background-color: ;
}
#mainwrapper .box .cap6{
    background-color: ;
}

#mainwrapper .box .bg-block-1{
  background-image: url("5 pointed collar category image.jpeg");
  background-color: rgba(26, 167, 88, 1);
}
#mainwrapper .box .bg-block-2{
  background-image:url("foolscap7.png");
  background-color:#d5582c;
}
#mainwrapper .box .bg-block-3{
  background-image:url("EH0080072_CV.jpg");
  background-color:rgba(46, 123, 204,1);
}
#mainwrapper .box .bg-block-4{
  background-image:url("EH0080073_CV.jpg");
  background-color:rgba(46, 123, 204,1);
}
#mainwrapper .box .bg-block-5{
  background-image:url("EH0080068_CV.jpg");
  background-color:#009fb2;
}
```

```
#mainwrapper .box .bg-block-6{
  background-image:url("basilisk.jpg");
  background-color:rgba(26, 167, 88, 1);
}

#mainwrapper .credit {
    font-size: 10px;
    position: relative;
}

#mainwrapper .credit p {
    color: white;
}
```

upload.css

The code below determines the appearance of the upload image section.

```
html, body {
  padding: 5px;
  background: url(../bg.png) no-repeat center center fixed;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
  background-size: cover;
}

body {
  font-family: sans-serif;
  background-color: #eeeeee;
}

.left-side {
```

```
    position: relative;
    float: left;
    width: 40%;
    overflow: auto;
}

/* Upload picture */
.file-upload {
  background-color: transparent;
  float: center;
  margin-top: 0px;
  position: relative;
}

.file-upload-btn {
  margin-left: 25%;
  width: 50%;
  color: black;
  background: black;
  border: none;
  padding: 10px;
  border-radius: 4px;
  transition: all .2s ease;
  outline: none;
  text-transform: uppercase;
  font-weight: 700;
}

.file-upload-btn:hover {
  background: black;
  color: black;
```

```
  transition: all .2s ease;
  cursor: pointer;
}

.file-upload-btn:active {
  border: 0;
  transition: all .2s ease;
}

.file-upload-content {
  display: none;
  text-align: center;
}

.file-upload-input {
  position: absolute;
  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  outline: none;
  opacity: 0;
  cursor: pointer;
}

.image-upload-wrap {
  margin-top: 20px;
  border: 4px dashed #ffffff;
  position: relative;
}
```

```
.image-dropping, .image-upload-wrap:hover {
  border: 4px dashed #999999;
  color: #999999;
}

.image-title-wrap {
  padding: 0 15px 15px 15px;
  color: #222;
}

.drag-text {
  text-align: center;
}

.drag-text h3 {
  font-weight: 100;
  text-transform: uppercase;
  color: #fff;
  padding: 60px 0;
}

.file-upload-image {
  max-height: 700px;
  max-width: 600px;
  margin: auto;
  padding: 20px;
}

/* Remove image button */
.remove-image {
  width: 60%;
```

```
  margin: 0;
  color: #fff;
  background: #666666;
  border: none;
  padding: 10px;
  border-radius: 4px;
  transition: all .2s ease;
  outline: none;
  text-transform: uppercase;
  font-weight: 700;
}

.remove-image:hover {
  background: #666666;
  color: #ffffff;
  transition: all .2s ease;
  cursor: pointer;
}

.remove-image:active {
  border: 0;
  transition: all .2s ease;
}

/* Open image editor button */
.image-editor {
  width: 60%;
  margin: 0;
  color: #fff;
  background: #666666;
  border: none;
```

```
  padding: 10px;
  border-radius: 4px;
  transition: all .2s ease;
  outline: none;
  text-transform: uppercase;
  font-weight: 700;
}

.image-editor:hover {
  background: #666666;
  color: #ffffff;
  transition: all .2s ease;
  cursor: pointer;
}

.image-editor:active {
  border: 0;
  transition: all .2s ease;
  background: #666666;
}

.image-editor:focus {
  border: 0;
  transition: all .2s ease;
  background: #666666;
}

/* rotate button */
.rotate {
  width: 60%;
  margin: 0;
```

```
  color: #fff;
  background: #666666;
  border: none;
  padding: 10px;
  border-radius: 4px;
  transition: all .2s ease;
  outline: none;
  text-transform: uppercase;
  font-weight: 700;
}

.rotate:hover {
  background: #666666;
  color: #ffffff;
  transition: all .2s ease;
  cursor: pointer;
}

.rotate:active {
  border: 0;
  transition: all .2s ease;
  background: #666666;
}
```

3. Instructions about the js files

editor.js

```
// editing image via css properties
function editImage() {
```

The code below reads the option values from the slider.

```
    var gs = $("#gs").val(); // grayscale
    var blur = $("#blur").val(); // blur
    var br = $("#br").val(); // brightness
```

```
var ct = $("#ct").val(); // contrast
var opacity = $("#opacity").val(); //opacity
var invert = $("#invert").val(); //invert
var saturate = $("#saturate").val(); //saturate
var sepia = $("#sepia").val(); //sepia
```

The code below applies filter to the uploaded image based on the option values.

```
$("#imageContainer img").css("filter", 'grayscale(' + gs+

                '%) blur(' + blur +

                'px) brightness(' + br +

                '%) contrast(' + ct +

                '%) opacity(' + opacity +

                '%) invert(' + invert +

                '%) saturate(' + saturate +

                '%) sepia(' + sepia + '%)');

$("#imageContainer img").css("-webkit-filter", 'grayscale(' + gs+

                '%) blur(' + blur +

                'px) brightness(' + br +

                '%) contrast(' + ct +
```

```
                    '%) opacity(' + opacity +

                    '%) invert(' + invert +

                    '%) saturate(' + saturate +

                    '%) sepia(' + sepia + '%)');
}
```

The purpose of the code below is that when sliders change image will be updated via editImage() function

```
$("input[type=range]").change(editImage).mousemove(editImage);
```

The code below is used to reset sliders back to their original values on press of 'reset'

```
$('#imageEditor').on('reset', function () {
        setTimeout(function() {
                editImage();
        },0);
});
```

The code below is the flipv function. The function changes the flipping status of the image based on the current status of the image.

```
function flipv() {
  var img = $('#imageContainer img');
   if(img.hasClass('f1')){
      img.attr('class','f4');
   }
   else if(img.hasClass('f4')){
      img.attr('class','f1');
```

```
    }
    else if(img.hasClass('f3')){
        img.attr('class','f2');
    }
    else if(img.hasClass('f2')){
        img.attr('class','f3');
    }
    else {
        img.attr('class','f4');
    }
};
```

The code below is the fliph function. The function changes the flipping status of the image based on the current status of the image.

```
function fliph() {
  var img = $('#imageContainer img');
    if(img.hasClass('f1')){
        img.attr('class','f2');
    }
    else if(img.hasClass('f2')){
        img.attr('class','f1');
    }
    else if(img.hasClass('f3')){
        img.attr('class','f4');
    }
    else if(img.hasClass('f4')){
        img.attr('class','f3');
    }
    else {
        img.attr('class','f2');
```

```
    }
};
```

main.js

```
/*
Reference: http://jsfiddle.net/BB3JK/47/
*/

$('select').each(function(){
    var $this = $(this), numberOfOptions = $(this).children('option').length;

    $this.addClass('select-hidden');
    $this.wrap('<div class="select"></div>');
    $this.after('<div class="select-styled"></div>');

    var $styledSelect = $this.next('div.select-styled');
    $styledSelect.text($this.children('option').eq(0).text());

    var $list = $('<ul />', {
        'class': 'select-options'
    }).insertAfter($styledSelect);

    for (var i = 0; i < numberOfOptions; i++) {
        $('<li />', {
            text: $this.children('option').eq(i).text(),
            rel: $this.children('option').eq(i).val()
        }).appendTo($list);
    }

    var $listItems = $list.children('li');
```

```
    $styledSelect.click(function(e) {
        e.stopPropagation();
        $('div.select-styled.active').each(function(){
            $(this).removeClass('active').next('ul.select-options').hide();
        });
        $(this).toggleClass('active').next('ul.select-options').toggle();
    });

    $listItems.click(function(e) {
        e.stopPropagation();
        $styledSelect.text($(this).text()).removeClass('active');
        $this.val($(this).attr('rel'));
        $list.hide();
        //console.log($this.val());
    });

    $(document).click(function() {
        $styledSelect.removeClass('active');
        $list.hide();
    });

});
```

The code below is used to determine the option that users choose from the option box. Different option will lead to different pages.

```
function finalLink() {
    var sel = document.getElementById('finalSelection');

    if (sel.options[sel.selectedIndex].value == 'kba') {
        location.href='kba/index.html';
    }
```

```
    else {
        location.href='kbb/index.html';
    }
};
```

upload.js

The code below is used to upload the image to the webpage.

```
function readURL(input) {
  if (input.files && input.files[0]) {
    var reader = new FileReader();

    reader.onload = function(e) {
      $('.image-upload-wrap').hide();

      $('.file-upload-image').attr('src', e.target.result);
      $('.file-upload-content').show();

      $('.image-title').html(input.files[0].name);
    };

    reader.readAsDataURL(input.files[0]);
  }

  else {
    removeUpload();
  }
}
```

The code below enables the users to drag and drop images to the upload image block.

```
$('.image-upload-wrap').bind('dragover', function () {
```

```
                $('.image-upload-wrap').addClass('image-dropping');
        });

$('.image-upload-wrap').bind('dragleave', function () {
    $('.image-upload-wrap').removeClass('image-dropping');
});
```

The code below is used save the upload image to the local storage.

```
function save() {
    bannerImage = document.getElementById('uploadedImage');
    imgData = getBase64Image(bannerImage);
    localStorage.setItem("imgData", imgData);
}
```

The code below is used to convert the upload image to a base64 image.

```
function getBase64Image(img) {
    var canvas = document.createElement("canvas");
    canvas.width = img.width;
    canvas.height = img.height;

    var ctx = canvas.getContext("2d");
    ctx.drawImage(img, 0, 0);

    var dataURL = canvas.toDataURL("image/png");

    return dataURL.replace(/^data:image\/(png|jpg);base64,/, "");
}
```

The code below is used to remove the image from the page.

```
function removeUpload() {
  $('.file-upload-input').replaceWith($('.file-upload-input').clone());
  $('.file-upload-content').hide();
  $('.image-upload-wrap').show();
```

```
}
```

The code below is used to open the image editor block.

```
function showEditor() {
   document.getElementById('editor').style.display = "block";
   document.getElementById('flipButton').style.display = "block";
```