Image Manipulator

Jash Nimje

INTRODUCTION

“Image Manipulator” is a image editing website that is aimed at instant editing and adding filters to our image. This is a web-based application to add all types of filters to our image of any size at one go. Image Manipulator aims to help people beautify their image quickly and securely by making all the changes locally without uploading image to the server. Every image uploaded recides in the local storage of the clients phone or pc. Thus making it secure and fully functional image editor. Image Manipulator aims to be a one stop guide for all the image editing needs. Image Manipulator has various features to cater different people's needs. It offers a green screen editing tool to superimpose other image onto the current image where green screen is visible. It offers all the color options to add as a filter onto any image of any size securely. It is a responsive and dynamic website that supports all types of devices ranging from pc, laptops and various sizes of mobile phones to cater a good user experience to the people using it.

ABSTRACT

The recent past showed a greater interest in image editing tools and techniques. Now-adays there are many websites that have emerged to offer editing tools. A user finds it a bit tedious to simply a add a simple filter to the image. It is way too many steps to simply apply a filter to any image. Using this website one can easily add filters to the image quickly and securely using Image Manipulator website. Inorder to do this one used to import project and then add multiple layers and filters and then export and that makes it too much cumbersome if a user wants to quickly add a simple filter. This website has green screen filter inorder to quickly edit out background of an image with green screen and add another image in that place. This makes it handy for the user to quickly replace green screen with their own choice of image.

The software and technique used to create this website are VS Code, HTML, CSS, PHP, JS and Chrome browser for testing purposes. The main motto is to attain user satisfaction who just like to add filters quickly without worrying about privacy concern. This website is responsive, thus it is extremely convenient for all types of devices.

WEBSITE

FRAMEWORK

A framework is a standardized set of concepts, practices and criteria for dealing with a common type of problem, which can be used as a reference to help us approach and resolve new problems of a similar nature.

In the world of web design, to give a more straightforward definition, a framework is defined as a package made up of a structure of files and folders of standardized code (HTML, CSS, JS documents etc.) which can be used to support the development of websites, as a basis to start building a site.

Most websites share a very similar (not to say identical) structure. The aim of frameworks is to provide a common structure so that developers don't have to redo it from scratch and can reuse the code provided. In this way, frameworks allow us to cut out much of the work and save a lot of time.

Website
All the elements in our website are made using PHP, HTML, CSS and Javascript without using any external frameworks.

CSS

CSS comes in three types:

- In a separate file (external)
- At the top of a web page document (internal)
- Right next to the text it decorates (inline)

External style sheets are separate files full of CSS instructions (with the file extension .css). When any web page includes an external stylesheet, its look and feel will be controlled by this CSS file (unless you decide to override a style using one of these next two types). This is how you change a whole website at once. And that's perfect if you want to keep up with the latest fashion in web pages without rewriting every page!

Internal styles are placed at the top of each web page document, before any of the content is listed. This is the next best thing to external, because they're easy to find, yet allow you to 'override' an external style sheet -- for that special page that wants to be a nonconformist!

Inline styles are placed right where you need them, next to the text or graphic you wish to decorate. You can insert inline styles anywhere in the middle of your

HTML code, giving you real freedom to specify each web page element. On the other hand, this can make maintaining web pages a real chore!

JAVASCRIPT

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

PHP

PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic.

MODULES

- **Home**: It is a welcome page to the website. It pops up the guide on how to use the website and how to add filters and save the images that are edited in this website.
- **Filter**: Here there are many filters in the default option that can be added to the image also one can add custom filters as well using colorize option.
- **Green Screen**: In this module users can replace green screen with other image of users choice and can easily download the superimposed image and edit out filters.

CODE

Script.js

This code demonstrates how filters are made using JavaScript and its formula and working.

```
// Hide Color Picker byDefault
document.getElementById("colorPicked").style.visibility = "hidden";

// Navigation
function openNav() {
  document.getElementById("navbar").style.width = "250px";
}

function closeNav() {
  document.getElementById("navbar").style.width = "0";
}

/* Filters */
var image = null;
var fimage = null;
var canvas = document.getElementById("display");

function upload() {
  var file = document.getElementById("upload");
  image = new SimpleImage(file);
  fimage = new SimpleImage(file);
  image.drawTo(canvas);
}

// Image Size Details
function displayDetails() {
  if (imageIsLoaded(image)) {
    var details = "<b>Image Size:</b> ";
    var width = image.getWidth();
    var height = image.getHeight();
    details = details + width + " x " + height;
    var p = document.getElementById("imgDetails");
    p.innerHTML = details;
    p.style.visibility = "visible";
```

```
  }
}

function showHide() {
  var selectedValue = filter.value;
  if (selectedValue === "colorize") {
    document.getElementById("colorPicked").style.visibility = "visible";
  } else {
    document.getElementById("colorPicked").style.visibility = "hidden";
  }
}

// Filter Dropdown Menu
function filterSelect() {
  var e = document.getElementById("filter");
  var selected = e.value;
  fimage = new SimpleImage(image);
  if (imageIsLoaded(fimage)) {
    if (selected === "grayscale") {
      grayscale();
    } else if (selected === "sepia") {
      sepia();
    } else if (selected === "rainbow") {
      rainbow();
    } else if (selected === "red") {
      red();
    } else if (selected === "blur") {
      blur();
    } else if (selected === "windowPane") {
      windowPane();
    } else if (selected === "colorize") {
      colorize();
    } else {
      alert("Select Filter");
    }

    var canvas = document.getElementById("display");
    fimage.drawTo(canvas);
  }
}
```

```
function imageIsLoaded(img) {
  if (img == null || !img.complete()) {
    alert("Image not loaded");
    return false;
  } else {
    return true;
  }
}

function resetImg() {
  if (imageIsLoaded(image)) {
    image.drawTo(canvas);
    fimage = new SimpleImage(image);
  }
}

// grayscale
function grayscale() {
  for (var pixel of fimage.values()) {
    var avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue()) / 3;
    pixel.setRed(avg);
    pixel.setGreen(avg);
    pixel.setBlue(avg);
  }
}

// Sepia
function sepia() {
  for (var pixel of fimage.values()) {
    r = pixel.getRed();
    g = pixel.getGreen();
    b = pixel.getBlue();
    newRed = 0.393 * r + 0.769 * g + 0.189 * b;
    newGreen = 0.349 * r + 0.686 * g + 0.168 * b;
    newBlue = 0.272 * r + 0.534 * g + 0.131 * b;
    if (newRed > 255) {
      newRed = 255;
    } else if (newGreen > 255) {
      newGreen = 255;
```

```
    } else if (newBlue > 255) {
      newBlue = 255;
    }

    pixel.setRed(newRed);
    pixel.setGreen(newGreen);
    pixel.setBlue(newBlue);
  }
}

// Rainbow
function rainbow() {
  var height = fimage.getHeight();
  for (var pixel of fimage.values()) {
    var y = pixel.getY();
    var avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue()) / 3;
    if (y < height / 7) {
      //red
      if (avg < 128) {
        pixel.setRed(2 * avg);
        pixel.setGreen(0);
        pixel.setBlue(0);
      } else {
        pixel.setRed(255);
        pixel.setGreen(2 * avg - 255);
        pixel.setBlue(2 * avg - 255);
      }
    } else if (y < (height * 2) / 7) {
      //orange
      if (avg < 128) {
        pixel.setRed(2 * avg);
        pixel.setGreen(0.8 * avg);
        pixel.setBlue(0);
      } else {
        pixel.setRed(255);
        pixel.setGreen(1.2 * avg - 51);
        pixel.setBlue(2 * avg - 255);
      }
    } else if (y < (height * 3) / 7) {
      //yellow
```

```
    if (avg < 128) {
      pixel.setRed(2 * avg);
      pixel.setGreen(2 * avg);
      pixel.setBlue(0);
    } else {
      pixel.setRed(255);
      pixel.setGreen(255);
      pixel.setBlue(2 * avg - 255);
    }
  } else if (y < (height * 4) / 7) {
    //green
    if (avg < 128) {
      pixel.setRed(0);
      pixel.setGreen(2 * avg);
      pixel.setBlue(0);
    } else {
      pixel.setRed(2 * avg - 255);
      pixel.setGreen(255);
      pixel.setBlue(2 * avg - 255);
    }
  } else if (y < (height * 5) / 7) {
    //blue
    if (avg < 128) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(2 * avg);
    } else {
      pixel.setRed(2 * avg - 255);
      pixel.setGreen(2 * avg - 255);
      pixel.setBlue(255);
    }
  } else if (y < (height * 6) / 7) {
    //indigo
    if (avg < 128) {
      pixel.setRed(0.8 * avg);
      pixel.setGreen(0);
      pixel.setBlue(2 * avg);
    } else {
      pixel.setRed(1.2 * avg - 51);
      pixel.setGreen(2 * avg - 255);
```

```
      pixel.setBlue(255);
    }
   } else {
    //violet
    if (avg < 128) {
     pixel.setRed(1.6 * avg);
     pixel.setGreen(0);
     pixel.setBlue(1.6 * avg);
    } else {
     pixel.setRed(0.4 * avg + 153);
     pixel.setGreen(2 * avg - 255);
     pixel.setBlue(0.4 * avg + 153);
    }
   }
  }
}

// Red
function red() {
 for (var pixel of fimage.values()) {
  var avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue()) / 3;
  if (avg < 128) {
   pixel.setRed(2 * avg);
   pixel.setGreen(0);
   pixel.setBlue(0);
  } else {
   pixel.setRed(255);
   pixel.setGreen(2 * avg - 255);
   pixel.setBlue(2 * avg - 255);
  }
 }
}

// Blur
function blur() {
 for (var pixel of fimage.values()) {
  var x = pixel.getX();
  var y = pixel.getY();
  var num = Math.random();
  if (num < 0.5) {
```

```
      fimage.setPixel(x, y, pixel);
    } else {
      fimage.setPixel(x, y, getRandomPixel(x, y, 20));
    }
  }
}
function getRandomPixel(x, y, distance) {
  var randomX = x + Math.random() * distance;
  var randomY = y + Math.random() * distance;

  if (randomX < 0) {
    randomX = 0;
  }
  if (randomX > fimage.getWidth()) {
    randomX = fimage.getWidth() - 1;
  }
  if (randomY < 0) {
    randomY = 0;
  }
  if (randomY > fimage.getHeight()) {
    randomY = fimage.getHeight() - 1;
  }
  return fimage.getPixel(randomX, randomY);
}

//Window Pane
function windowPane() {
  for (var pixel of fimage.values()) {
    var x = pixel.getX();
    var y = pixel.getY();

    if (
      pixel.getX() < 40 ||
      (pixel.getX() < fimage.getWidth() &&
        pixel.getX() > fimage.getWidth() - 40) ||
      pixel.getY() < 40 ||
      (pixel.getY() < fimage.getHeight() &&
        pixel.getY() > fimage.getHeight() - 40) ||
      (pixel.getX() < fimage.getWidth() / 4 &&
        pixel.getX() > fimage.getWidth() / 4 - 30) ||
```

```
      (pixel.getX() < fimage.getWidth() / 2 &&
        pixel.getX() > fimage.getWidth() / 2 - 30) ||
      (pixel.getX() < (fimage.getWidth() * 3) / 4 &&
        pixel.getX() > (fimage.getWidth() * 3) / 4 - 30) ||
      (pixel.getY() < fimage.getHeight() / 2 &&
        pixel.getY() > fimage.getHeight() / 2 - 30)
    ) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(0);
    }
  }
}

// Colorize
function parseColor(RGB) {
  var color = document.getElementById("colorPicked").value;
  return parseInt(color.substr(RGB, 2), 16);
}
function calculateLow(color) {
  return color / 127.5;
}
function calculateHigh(color) {
  return 2 - color / 127.5;
}

function colorizePixel(avg, red, green, blue, pixel) {
  var redL = calculateLow(red);
  var redH = calculateHigh(red);
  var greenL = calculateLow(green);
  var greenH = calculateHigh(green);
  var blueL = calculateLow(blue);
  var blueH = calculateHigh(blue);
  if (avg < 128) {
    pixel.setRed(redL * avg);
    pixel.setGreen(greenL * avg);
    pixel.setBlue(blueL * avg);
  } else {
    pixel.setRed(redH * avg + 2 * red - 255);
    pixel.setGreen(greenH * avg + 2 * green - 255);
```

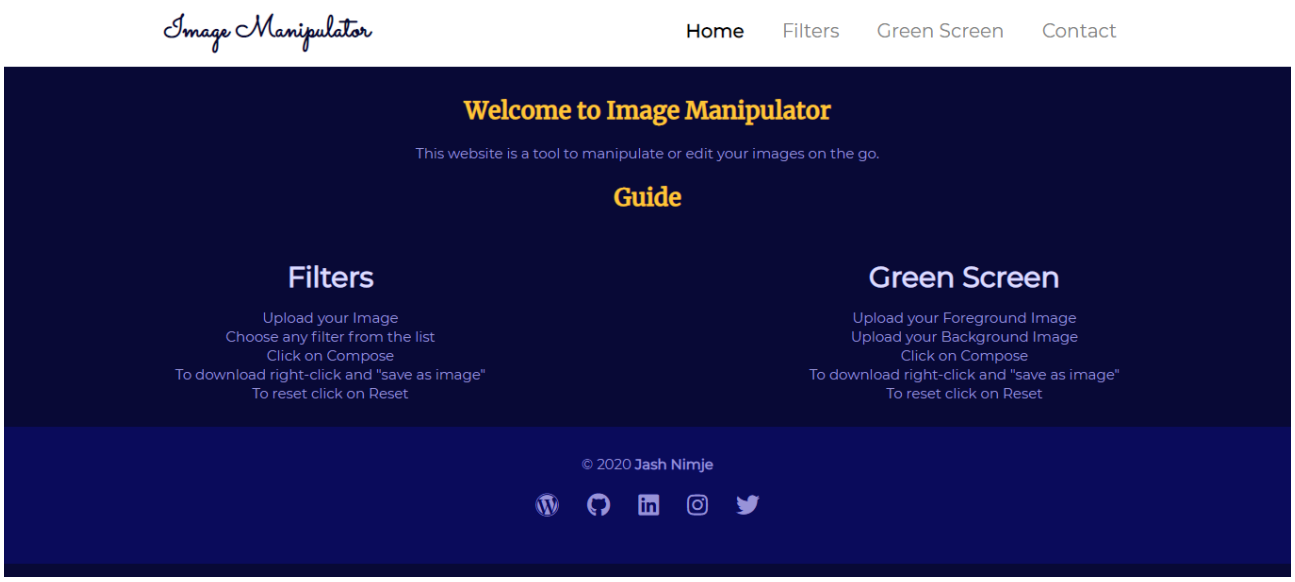
```
    pixel.setBlue(blueH * avg + 2 * blue - 255);
  }
}

//document.getElementById("element").style.display = "none";
function colorize() {
  for (var pixel of fimage.values()) {
    var avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue()) / 3;
    var red = parseColor(1);
    var green = parseColor(3);
    var blue = parseColor(5);
    colorizePixel(avg, red, green, blue, pixel);
  }
}

// Tabs
function features(evt, featureName) {
  var i, tabcontent, tablinks;

  // Get all elements with class="tabcontent" and hide them
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }

  // Get all elements with class="tablinks" and remove the class "active"
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }

  // Show the current tab, and add an "active" class to the button that opened the tab
  document.getElementById(featureName).style.display = "block";
  evt.currentTarget.className += " active";
}

/* Green Screen */
var fgImage = null;
var bgImage = null;
var fgCanvas;
```

```
var bgCanvas;

function loadForegroundImage() {
  var file = document.getElementById("fgfile");
  fgImage = new SimpleImage(file);
  fgCanvas = document.getElementById("fgcan");
  fgImage.drawTo(fgCanvas);
}

function loadBackgroundImage() {
  var file = document.getElementById("bgfile");
  bgImage = new SimpleImage(file);
  bgCanvas = document.getElementById("bgcan");
  bgImage.drawTo(bgCanvas);
}

function createComposite() {
  // this function creates a new image with the dimensions of the foreground image and
returns the composite green screen image
  var output = new SimpleImage(fgImage.getWidth(), fgImage.getHeight());
  var greenThreshold = 240;
  for (var pixel of fgImage.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (pixel.getGreen() > greenThreshold) {
      //pixel is green, use background
      var bgPixel = bgImage.getPixel(x, y);
      output.setPixel(x, y, bgPixel);
    } else {
      //pixel is not green, use foreground
      output.setPixel(x, y, pixel);
    }
  }
  return output;
}

function doGreenScreen() {
  //check that images are loaded
  if (fgImage == null || !fgImage.complete()) {
    alert("Foreground image not loaded");
```

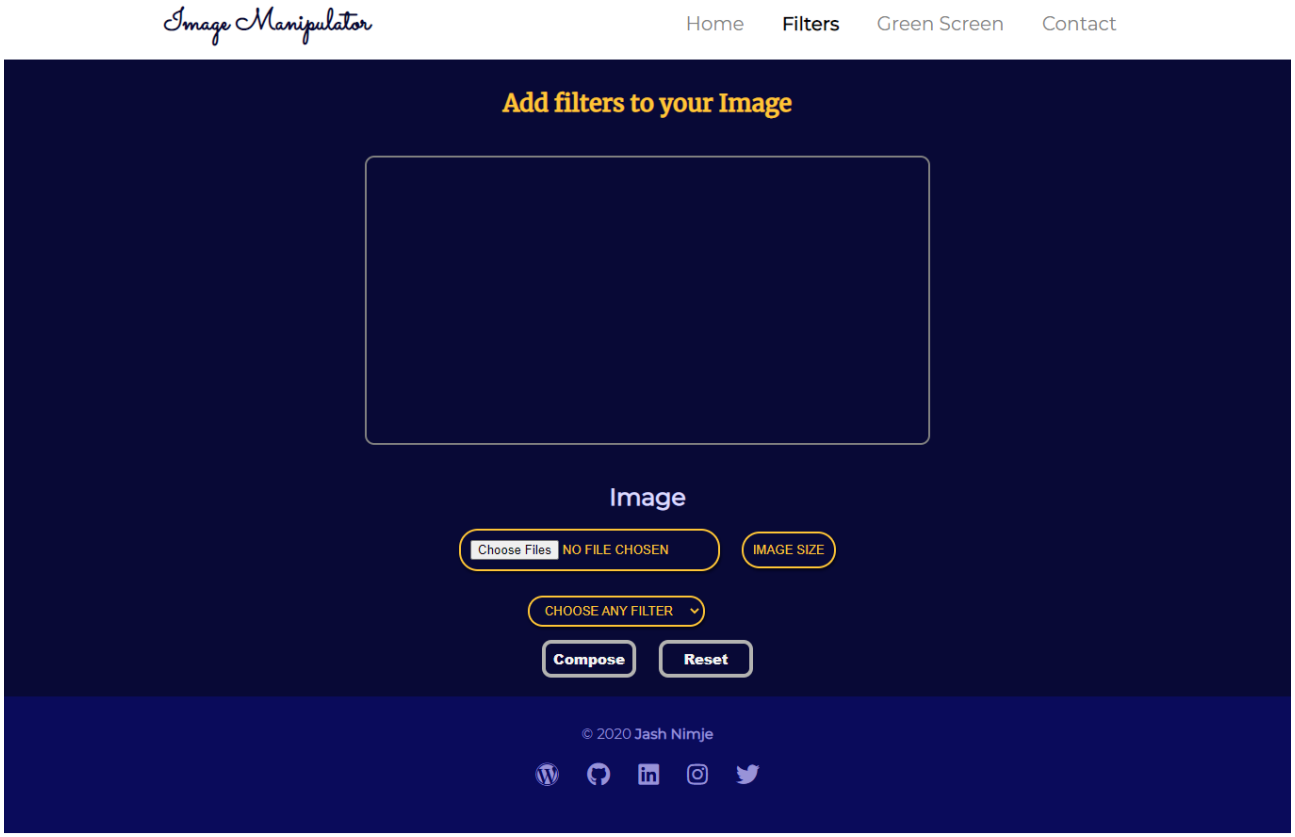
```
  }
  if (bgImage == null || !bgImage.complete()) {
    alert("Background image not loaded");
  }
  // clear canvases
  clearCanvas();
  // call createComposite, which does green screen algorithm and returns a composite
image
  var finalImage = createComposite();
  finalImage.drawTo(fgCanvas);
}

function clearCanvas() {
  doClear(fgCanvas);
  doClear(bgCanvas);
}

function doClear(canvas) {
  var context = canvas.getContext("2d");
  context.clearRect(0, 0, canvas.width, canvas.height);
}
```

SCREENSHOTS

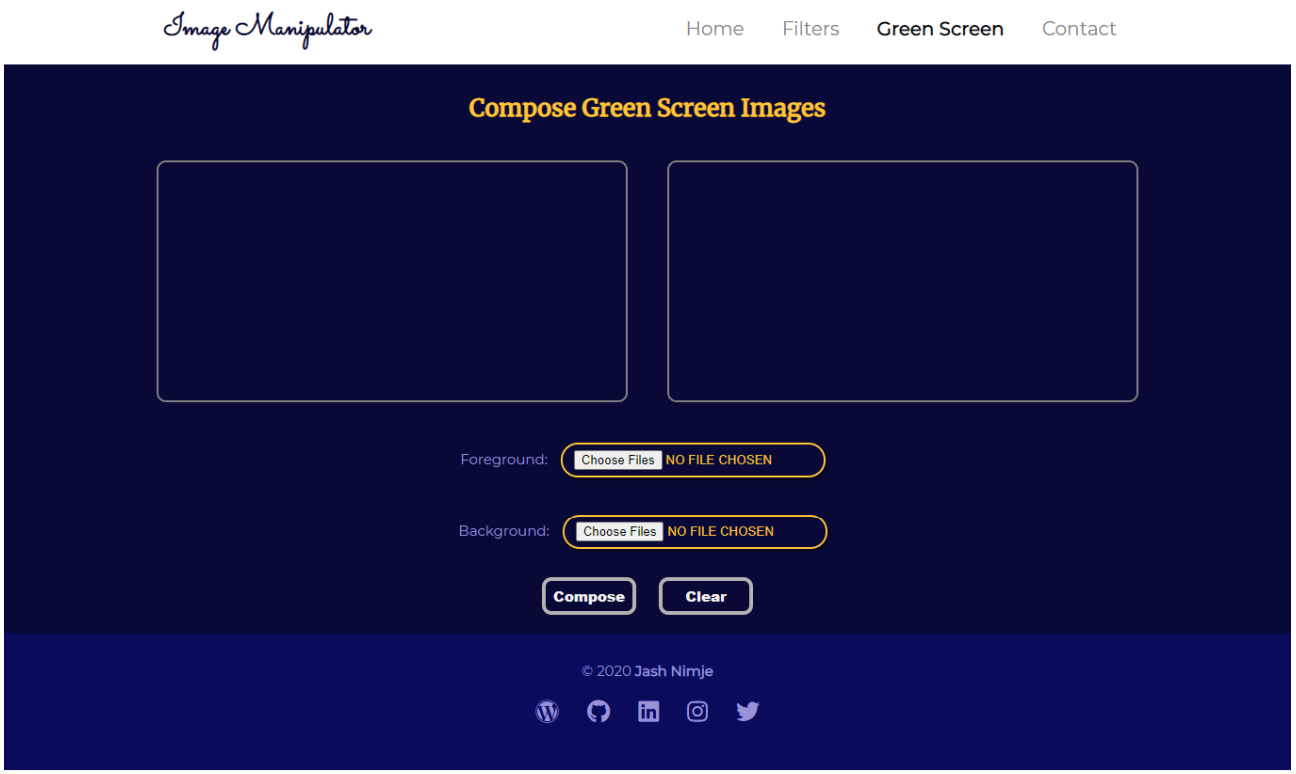
Homepage:



Filter:



Green Screen:



CONCLUSION

Through this project we took a step towards minimizing the efforts the user has to take to go to different websites for simple editing of filters and green screen editing. In this, a user will select a filter that they want to add to their image. We have brought the green screen editing as well in one website so that users can edit the green screen background of an image and get perfect image edit. Image preview is provided which makes it easy for the user to choose the filter for an image and get a good idea of the image. Finally, the goal of the project is to make an effective website for adding filters and edit image quickly and securely in an efficient manner across device.