**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**KUMASI**

COLLEGE OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

PROJECT DOCUMENTATION



**TOPIC: IMAGE EDITOR WEB APPLICATION USING HTML, CSS AND JAVASCRIPT**

**NORVIDEKA COURAGE - 4218620**

# **SUPERVISED BY:** **DR. (MR.) KWABENA OWUSU AGYEMAN**

**DOCUMENTATION ON MY PROJECT**

**Introduction**

**Key Features**

**Project Structure**

HTML Structure

CSS Styling

JavaScript Functionality

**Usage Instructions**

**Conclusion**

**INTRODUCTION**

The proposed project is to develop an Image Editor Web Application that empowers users to upload images, apply filters and transformations, preview changes in real-time, and save their edited images. The application will be implemented using HTML, CSS, and JavaScript, ensuring a user-friendly interface, responsive design, and seamless functionality.

**OBJECTIVES**

* Create an interactive web-based platform that allows users to edit images using various filters and transformations.
* Provide users with real-time previews of filter effects and transformations applied to the images.
* Implement responsive design to ensure optimal user experience across different devices and screen sizes.

Enable users to upload their images, apply filters, rotate, flip, reset changes, and save their edited images.

**KEY FEATURES**

* Filter Selection: Users can choose from a set of filters (brightness, saturation, inversion, grayscale) to apply to their images.
* Real-time Preview: Users will witness instant updates in the preview image as they adjust filters and transformations.
* Rotation and Flipping: The application will allow users to rotate images left or right by 90 degrees and flip them horizontally or vertically.
* Reset Filters: A "Reset Filters" button will reset all modifications and restore the image to its original state.
* Image Upload: Users can upload their images to edit within the application.
* Save Edited Image: The edited images can be saved by users after applying filters and transformations.
* Responsive Design: The application's layout and functionalities will adapt to various screen sizes for a consistent experience.

**TECHNOLOGIES USED**

* HTML: For creating the structure and content of the application.
* CSS: For styling the user interface, ensuring a visually appealing design.
* JavaScript: For adding interactivity, applying filters, handling transformations, and providing real-time previews.
* Canvas API: To create and save edited images.

# 

**PROJECT STRUCTURE**

**HTML STRUCTURE**

The HTML structure of the application is well-organized, with separate sections for the header, filter panel, image preview, and controls. The structure ensures clear separation of different components and enhances the user experience.

Meta Information and Title:

<META CHARSET="UTF-8"><TITLE>iMAGE eDITOR IN jAVAsCRIPT | cODINGnEPAL</TITLE>

These meta tags define the character encoding and title of the web page. The title will be displayed on the browser tab or window.

Viewport Meta Tag:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

This meta tag ensures that the web page is properly scaled to fit different screen sizes and devices. It's crucial for creating a responsive design.

CSS and Font Links:

<link rel="stylesheet" href="style.css"><link rel="stylesheet" href="https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css"><link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css"/>

These <link> tags link to external CSS stylesheets that provide styling for the application. style.css is the local stylesheet, while the other two links bring in icon fonts (Boxicons and Font Awesome) for the UI.

Container Structure:

<div class="container disable">

<!-- ... --></div>

This <div> element with the class "container" wraps the entire content of the application. The "disable" class is initially added to disable user interactions until an image is loaded.

Filter Buttons:

<button id="brightness" class="active">Brightness</button><button id="saturation">Saturation</button><button id="inversion">Inversion</button><button id="grayscale">Grayscale</button>

These <button> elements represent the filter options available to the user. They have unique IDs for JavaScript to identify them, and the "active" class initially highlights the "Brightness" filter as active.

Slider for Filters:

<input type="range" value="100" min="0" max="200">

This <input> element of type "range" allows users to adjust the intensity of the selected filter. It has a range from 0 to 200 and an initial value of 100.

Rotate and Flip Buttons:

<button id="left"><i class="fa-solid fa-rotate-left"></i></button><button id="right"><i class="fa-solid fa-rotate-right"></i></button><button id="horizontal"><i class='bx bx-reflect-vertical'></i></button><button id="vertical"><i class='bx bx-reflect-horizontal'></i></button>

These <button> elements use icons (from Font Awesome and Boxicons) to represent rotation and flip options for the image.

Preview Image:

<div class="preview-img">

<img src="image-placeholder.svg" alt="preview-img"></div>

This <div> contains the preview image element, which initially displays a placeholder image. The <img> element will later display the user-uploaded or edited image.

File Input and Control Buttons:

<input type="file" class="file-input" accept="image/\*" hidden><button class="choose-img">Choose Image</button><button class="save-img">Save Image</button>

The <input> element of type "file" serves as a hidden file input for uploading images. The "accept" attribute restricts it to image files only. The "Choose Image" and "Save Image" buttons allow users to trigger image upload and save actions, respectively.

**CSS STYLING**

The CSS styles are defined in a separate stylesheet (style.css). The application employs the "Poppins" Google font for a visually appealing and readable text. The styles are carefully crafted to create a clean and intuitive interface, with responsive design adjustments for various screen sizes.

Global Reset and Font Settings:

\*{

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

This CSS snippet is a global reset that removes default margins and paddings from all elements. It sets the box-sizing property to "border-box," ensuring that padding and border don't affect an element's total width and height. The font-family is set to 'Poppins' with a fallback to the generic 'sans-serif' font.

Responsive Flex Layout and Background:

body{

display: flex;

padding: 10px;

min-height: 100vh;

align-items: center;

justify-content: center;

background: #E3F2FD;

}

This styling configures the layout of the entire page. It uses a flex display to center the content both horizontally and vertically. The background property sets a light blue background color.

Container Styling:

.container{

width: 850px;

padding: 30px 35px 35px;

background: #fff;

border-radius: 10px;

box-shadow: 0 10px 20px rgba(0,0,0,0.1);

}

The .container class styles the main content container. It sets the width, padding, background color, border-radius (rounded corners), and box-shadow (gives a subtle shadow) to create a card-like appearance.

Button Styling:

.editor-panel button, .controls button{

outline: none;

/\* ... \*/

}

This CSS rule targets all buttons within the .editor-panel and .controls classes. It removes the default outline when buttons are clicked, improving the appearance.

Active Filter Button Styling

.filter button.active{

color: #fff;

border-color: #5372F0;

background: #5372F0;

}

This rule styles the active filter button. When a filter is selected, this styling changes the text color, border color, and background color to highlight the active filter.

Preview Image Styling:

.preview-img img{

max-width: 490px;

max-height: 335px;

width: 100%;

height: 100%;

object-fit: contain;

}

The CSS here styles the preview image. It sets maximum width and height limits, ensuring the image fits within those dimensions. The object-fit property ensures that the image retains its aspect ratio while fitting within the specified dimensions.

Media Queries for Responsiveness:

@media screen and (max-width: 760px) {

/\* ... \*/

}@media screen and (max-width: 500px) {

/\* ... \*/

}

These media queries adjust the layout and styling for different screen sizes. The CSS within these queries adapts the design to offer an optimal experience on smaller screens.

**JAVASCRIPT FUNCTIONALITY**

The JavaScript code powers the interactive functionalities of the application. It establishes event listeners for filter buttons, sliders, rotation and flip buttons, file input changes, and more. The code ensures smooth synchronization between user interactions and visual changes on the preview image. The application uses the Canvas API to create and save edited images.

Load Image Function:

const loadImage = () => {

// ...

}

This function is called when an image is selected using the file input. It loads the selected image into the preview area, sets up event listeners for user interactions, and enables the editing controls.

Apply Filter Function:

const applyFilter = () => {

// ...

}

This function applies the selected filters and transformations to the preview image. It modifies the CSS properties of the image element, such as rotation, scaling, and filter effects.

Filter Options Event Listeners:

filterOptions.forEach(option => {

option.addEventListener("click", () => {

// ...

});

});

These event listeners are set on each filter button. When a filter button is clicked, the active filter is highlighted, and the slider range and value are adjusted accordingly.

Update Filter Function:

const updateFilter = () => {

// ...

}

This function updates the value displayed on the slider and adjusts the respective filter variable (brightness, saturation, etc.) based on the slider's value. It also calls the applyFilter function to update the preview.

Rotate and Flip Options Event Listeners:

rotateOptions.forEach(option => {

option.addEventListener("click", () => {

// ...

});

});

Similar to filter options, these event listeners handle the user's selection of rotation and flipping options. The applyFilter function is called to apply these transformations.

Reset Filter Function:

const resetFilter = () => {

// ...

}

This function resets all applied filters, transformations, and variables to their default values. It also updates the UI to display the "Brightness" filter as active and calls applyFilter to reset the preview image.

Save Image Function:

const saveImage = () => {

// ...

}

This function uses the Canvas API to create an edited image with the applied filters and transformations. The canvas content is then converted into a downloadable image format.

Event Listeners and Input Changes:

filterSlider.addEventListener("input", updateFilter);

resetFilterBtn.addEventListener("click", resetFilter);

saveImgBtn.addEventListener("click", saveImage);

fileInput.addEventListener("change", loadImage);

chooseImgBtn.addEventListener("click", () => fileInput.click());

These event listeners capture user interactions and input changes. They execute the corresponding functions when users interact with filter sliders, reset filters, save image, choose image, and change the file input.

**USAGE INSTRUCTIONS**

* Choose Image: Click the "Choose Image" button to upload an image you want to edit.
* Apply Filters: Click on filter buttons (Brightness, Saturation, Inversion, Grayscale) to select a filter. Adjust the slider to control the intensity of the filter effect.
* Rotate and Flip: Use the rotation and flip buttons to adjust the orientation of the image.
* Reset Filters: Click "Reset Filters" to revert all changes to the original image.
* Save Image: After editing, click the "Save Image" button to download the edited image.

**CONCLUSION**

In conclusion, the Image Editor web application offers a user-friendly platform for editing images with a range of filters and transformations. Through a combination of HTML, CSS, and JavaScript, the application provides real-time previews, responsive design, and a seamless user experience. By following the usage instructions provided in this documentation, users can easily edit and save their images with desired effects.

**REFERENCES: <https://chat.openai.com>**

**GITHUB LINK: <https://github.com/cnorvideka6>**