IPCV PROJECT

EditMaster: Interactive Image Enhancement TooL

1.Abstract:

EditMaster: Interactive Image Enhancement Tool* is a web application that enables real-time image editing with features like filters, cropping, and effects. Designed for both novices and professionals, it offers instant feedback and high-quality results. Ideal for social media content creation and graphic design, EditMaster simplifies the editing process, making creativity accessible to all users.

2. Code:

PYTHON:

```
from flask import Flask, render_template, request, url_for
from PIL import Image, ImageEnhance, ImageFilter
import numpy as np
import os

app = Flask(_name_)
app.config['UPLOAD_FOLDER'] = 'static/uploads/'
app.config['PROCESSED_FOLDER'] = 'static/processed/'

# Ensure folders exist
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['PROCESSED_FOLDER'], exist_ok=True)

@app.route("/", methods=["GET"])
```

```
def index():
    return render_template("index.html")

@app.route("/upload", methods=["POST"])
def upload():
    image_file = request.files.get("image")
    if image_file:
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], image_file.filename)
        image_file.save(image_path)
        return render_template("index.html", image_url=url_for('static', filename="uploads/" +
image_file.filename))
    return render_template("index.html", error="No file selected.")

@app.route("/edit", methods=["POST"])
def edit_image():
    action = request.form.get("action")
    uploaded_images = os.listdir(app.config['UPLOAD_FOLDER'])
    if not uploaded_images:
        return render_template("index.html", error="No uploaded image found.")

    uploaded_image_path = os.path.join(app.config['UPLOAD_FOLDER'],
uploaded_images[0])
    image = Image.open(uploaded_image_path)
    processed_path = os.path.join(app.config['PROCESSED_FOLDER'],
"processed_image.jpg")

    try:

        if action == "grayscale":
            image = image.convert("L")
        elif action == "flip":
            image = image.transpose(Image.FLIP_LEFT_RIGHT)
        elif action == "rotate":
            image = image.rotate(90, expand=True)
        elif action == "sepia":
            sepia_image = ImageEnhance.Color(image).enhance(0.3)
            sepia_image = sepia_image.convert("RGB")
            image = sepia_image
        elif action == "blur":
            image = image.filter(ImageFilter.GaussianBlur(5))
        elif action == "crop":
            width, height = image.size
```

```
            image = image.crop((0, 0, width // 2, height // 2))
        elif action == "rgb_to_binary":
            image = np.array(image.convert("L"))  # Convert to grayscale first
            threshold = 128
            binary_image = (image > threshold) * 255  # Binarize
            image = Image.fromarray(binary_image.astype('uint8'))
        elif action == "rgb_to_index":
            image = np.array(image.convert("RGB"))
            indexed_image = np.dot(image, [0.2989, 0.587, 0.114])  # Simplified index
            indexed_image = (indexed_image / indexed_image.max() * 255).astype('uint8')
            image = Image.fromarray(indexed_image)
        elif action == "sharpen":
            image = image.filter(ImageFilter.SHARPEN)
        elif action == "edge_detect":
            image = image.filter(ImageFilter.FIND_EDGES)

        # Convert image to RGB if it is in RGBA or any mode that does not support JPEG
        if image.mode in ("RGBA", "LA"):
            image = image.convert("RGB")

        # Save the processed image
        image.save(processed_path, "JPEG")

    except Exception as e:
        return render_template("index.html", error=f"An error occurred: {e}")

    return render_template(
        "index.html",
        image_url=url_for('static', filename="uploads/" +
os.path.basename(uploaded_image_path)),
        processed_image_url=url_for('static', filename="processed/processed_image.jpg")
    )

if _name_ == "main":
    app.run(debug=True)

HTML:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Image Editor</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>
    <div class="container">
        <h1>💫Edit Master💫</h1>

        <!-- Uploaded Image Preview -->
        {% if image_url %}
        <h2>Uploaded Image:</h2>
        <img src="{{ image_url }}" alt="Uploaded Image" id="uploaded-image"
class="image-preview">
        {% endif %}

        <!-- Processed Image Preview -->
        {% if processed_image_url %}
        <h2>Processed Image:</h2>
        <img src="{{ processed_image_url }}" alt="Processed Image" id="processed-image"
class="image-preview">
        <br>
        <a href="{{ processed_image_url }}" download="processed_image.jpg">
            <button class="btn download-btn">Download Processed Image</button>
        </a>
        {% endif %}

        <!-- Upload Form -->
        <form action="/upload" method="POST" enctype="multipart/form-data"
id="upload-form">
            <label class="custom-file-upload">
                <input type="file" name="image" id="image-input" accept="image/*" required>
                Select File
            </label>
            <div id="preview-container">
                <img id="preview-image" alt="Preview will appear here" style="display: none;
max-width: 100%; border: 1px solid #ddd; border-radius: 5px;">
            </div>
            <button type="submit" class="btn upload-btn">Upload</button>
        </form>

        {% if image_url %}
        <!-- Edit Form -->
        <form action="/edit" method="POST">
```

```
        <label for="action" class="action-label">Select an Action:</label>
        <select name="action" id="action-select" class="custom-select" required>
          <option value="grayscale">Grayscale</option>
          <option value="flip">Flip Horizontally</option>
          <option value="rotate">Rotate 90°</option>
          <option value="sepia">Sepia</option>
          <option value="blur">Blur</option>
          <option value="crop">Crop</option>
          <option value="rgb_to_binary">RGB to Binary</option>
          <option value="rgb_to_index">RGB to Index</option>
          <option value="sharpen">Sharpen</option>
          <option value="edge_detect">Edge Detection</option>
        </select>

        <!-- Brightness Options -->
        <div id="brightness-options" class="action-options" style="display: none;">
          <label for="brightness-factor">Brightness Factor (0.0 to 2.0):</label>
          <input type="number" step="0.1" name="brightness_factor"
id="brightness-factor" value="1.0" min="0" max="2">
        </div>

        <!-- Contrast Options -->
        <div id="contrast-options" class="action-options" style="display: none;">
          <label for="contrast-factor">Contrast Factor (0.0 to 2.0):</label>
          <input type="number" step="0.1" name="contrast_factor" id="contrast-factor"
value="1.0" min="0" max="2">
        </div>

        <!-- Saturation Options -->
        <div id="saturation-options" class="action-options" style="display: none;">
          <label for="saturation-factor">Saturation Factor (0.0 to 2.0):</label>
          <input type="number" step="0.1" name="saturation_factor" id="saturation-factor"
value="1.0" min="0" max="2">
        </div>

        <button type="submit" class="btn">Apply</button>
      </form>
      {% endif %}
    </div>
    <script src="/static/script.js"></script>
</body>
</html>
```

SCRIPT:

```
// Preview image when selected for upload
document.getElementById("image-input").addEventListener("change", function (event) {
    const file = event.target.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = function (e) {
            const previewImage = document.getElementById("preview-image");
            previewImage.src = e.target.result;
            previewImage.style.display = "block";
        };
        reader.readAsDataURL(file);
    }
});

// Show/hide additional options based on selected action
const actionSelect = document.getElementById("action-select");
const actionOptions = document.querySelectorAll(".action-options");

actionSelect.addEventListener("change", function () {
    actionOptions.forEach(option => option.style.display = "none"); // Hide all options
    const selectedAction = actionSelect.value;
    const selectedOptions = document.getElementById(${selectedAction}-options);
    if (selectedOptions) {
        selectedOptions.style.display = "block"; // Show relevant options
    }
});
```

CSS:

```
<style>
/* Basic Reset */
*,
*::before,
*::after {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body Styling - Dark Theme */
body {
    font-family: 'Poppins', sans-serif;
    background: #181818; /* Dark background */
    color: #f0f0f0; /* Light text color */
    line-height: 1.6;
    padding: 50px 20px;
    text-align: center;
}

/* Container Styling */
.container {
    max-width: 950px;
    margin: 50px auto;
    background: #1e1e1e; /* Dark container background */
    border-radius: 15px;
    box-shadow: 0 15px 30px rgba(0, 0, 0, 0.4);
    padding: 40px;
    overflow: hidden;
    transition: all 0.3s ease;
}

.container:hover {
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.5);
}

/* Header Styling */
h1 {
    font-size: 2.5em; /* Adjust the size of the text */
```

```
    color: #0d0d0d; /* Tomato color for the header */
    margin-bottom: 15px; /* Space below the header */
    font-weight: 600; /* Semi-bold text */

    /* Block style properties */
    display: block; /* Ensures the header behaves like a block element */
    background-color: rgb(231, 115, 115); /* Dark background color */
    padding: 20px; /* Padding around the text */
    border-radius: 10px; /* Rounded corners */
    text-align: center; /* Center the text */
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3); /* Add a shadow effect */
}



    h2 {
        font-size: 1.6em;
        margin-bottom: 15px;
        color: #00bcd4; /* Light blue color for subheadings */
    }

    /* Image Preview */
    .image-preview {
        max-width: 100%;
        height: auto;
        border-radius: 12px;
        border: 3px solid #444;
        margin-top: 20px;
        box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
        transition: transform 0.3s ease, box-shadow 0.3s ease;
    }

    .image-preview:hover {
        transform: scale(1.05);
        box-shadow: 0 8px 20px rgba(0, 0, 0, 0.4);
    }

    #preview-image {
        max-width: 100%;
        border-radius: 8px;
        margin-top: 20px;
        transition: opacity 0.5s ease-in-out;
    }
```

```
#preview-image:empty {
    opacity: 0;
}

/* Form Styling */
form {
    margin-top: 30px;
    font-size: 1.1em;
}

form label {
    font-weight: 600;
    margin-bottom: 10px;
    display: block;
    color: #f0f0f0;
}

input[type="file"],
select {
    padding: 12px;
    border: 2px solid #444;
    border-radius: 8px;
    font-size: 1em;
    width: 100%;
    max-width: 320px;
    margin: 10px 0;
    background-color: #2a2a2a; /* Dark background for inputs */
    color: #f0f0f0; /* Light text */
    transition: border-color 0.3s ease, box-shadow 0.3s ease;
}

input[type="file"]:focus,
select:focus {
    border-color: #00bcd4; /* Light blue on focus */
    outline: none;
    box-shadow: 0 0 6px rgba(0, 188, 212, 0.6);
}

/* Button Styling */
button {
    background-color: #00bcd4; /* Light blue button */
```

```
    color: #181818; /* Dark text for contrast */
    padding: 14px 25px;
    border: none;
    border-radius: 8px;
    font-size: 1.2em;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.3s ease, box-shadow 0.3s ease;
    margin-top: 15px;
}

button:hover {
    background-color: #0097a7;
    transform: scale(1.05);
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
}

button:active {
    background-color: #007c85;
}

button:disabled {
    background-color: #777;
    cursor: not-allowed;
}

/* Download Button */
a button {
    background-color: #4caf50; /* Green button */
    margin-top: 20px;
}

a button:hover {
    background-color: #388e3c;
}

/* Select Menu Styling */
select {
    background-color: #333;
    color: #f0f0f0;
    font-size: 1em;
    cursor: pointer;
    transition: background-color 0.3s ease;
```

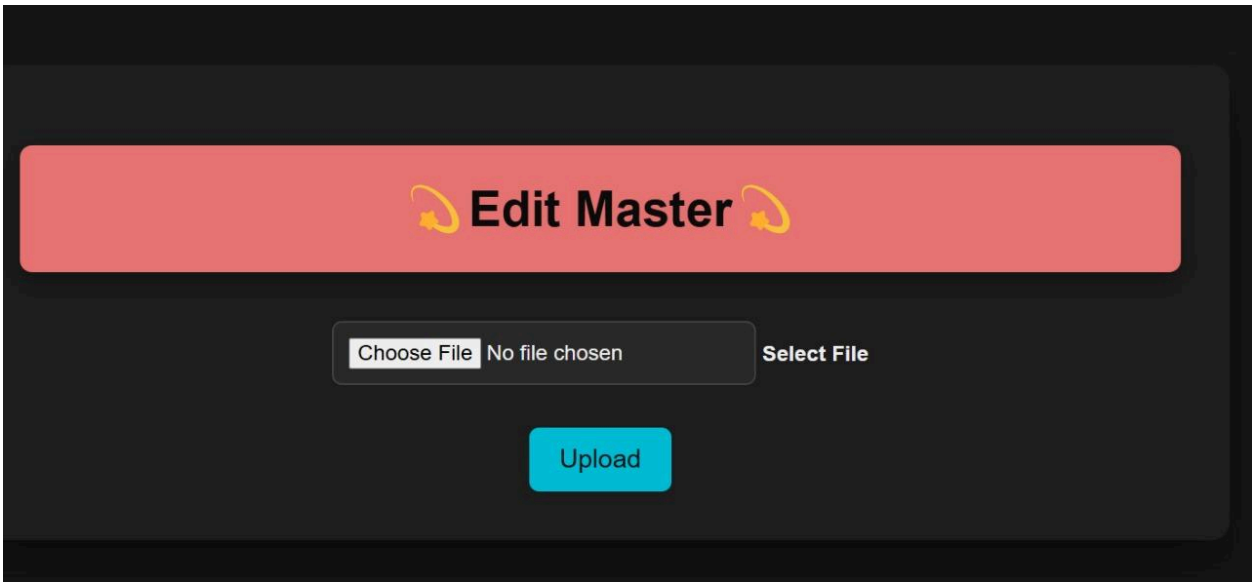
```
}

select:hover {
    background-color: #444;
}

/* Responsive Design */
@media (max-width: 768px) {
    .container {
        padding: 25px;
    }

    h1 {
        font-size: 2.2em;
    }

    .image-preview {
        margin-top: 10px;
    }

    button, input[type="file"], select {
        width: 100%;
        max-width: 100%;
    }
}

/* Loading Spinner */
#loading-spinner {
    display: none;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

#loading-spinner.show {
    display: block;
}

/* Modal for errors */
#error-modal {
    display: none;
```

```
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.8);
    justify-content: center;
    align-items: center;
}

#error-modal .modal-content {
    background: #2b2b2b;
    padding: 30px;
    border-radius: 12px;
    text-align: center;
    max-width: 400px;
    width: 90%;
}

#error-modal.show {
    display: flex;
}

#error-modal .modal-content p {
    margin-bottom: 20px;
    font-size: 1.3em;
    color: #e53935;
}

#error-modal .modal-content button {
    background-color: #e53935;
    color: #fff;
    padding: 10px 20px;
    border-radius: 8px;
    font-size: 1.1em;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

#error-modal .modal-content button:hover {
    background-color: #c62828;
}
```

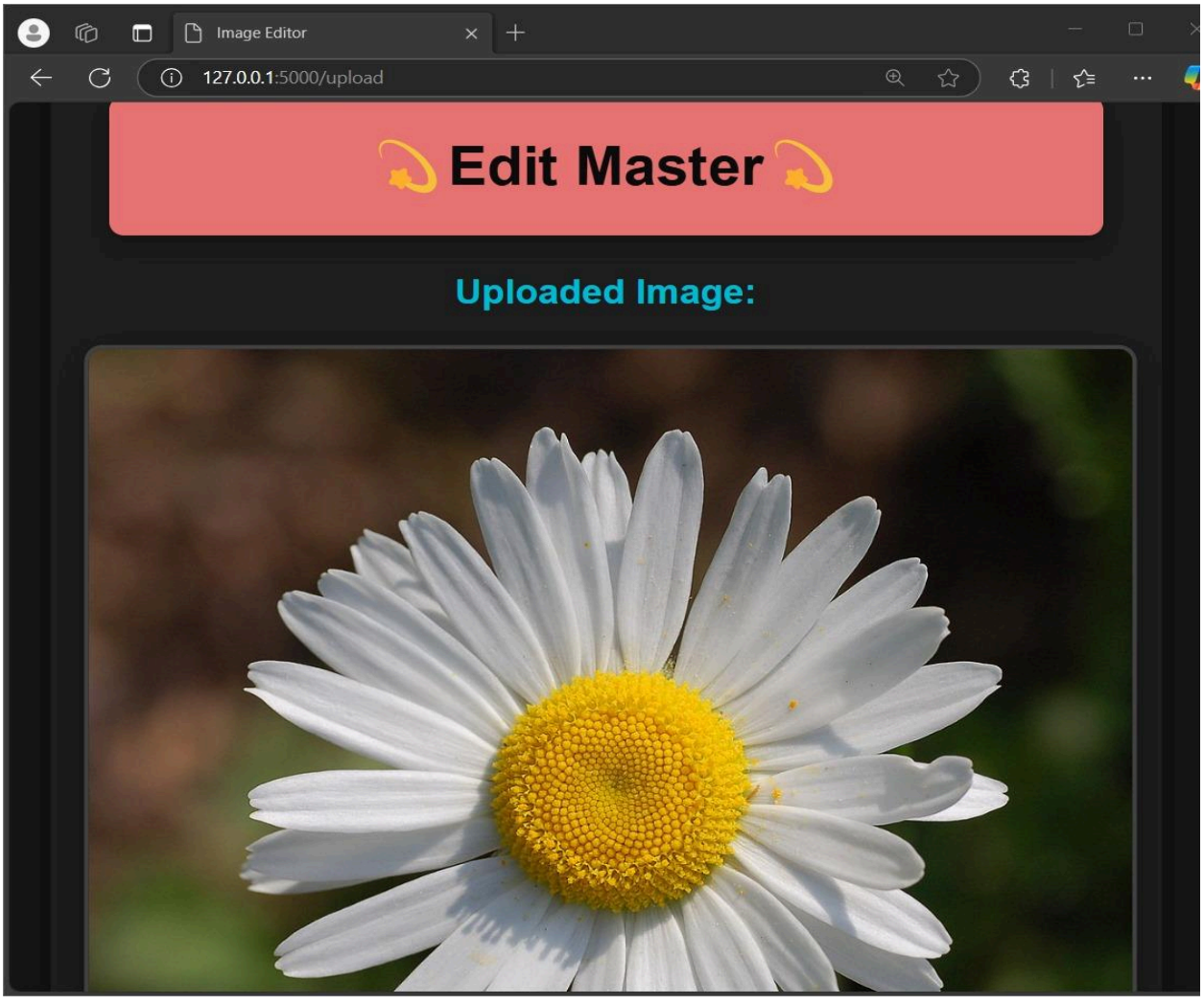
</style>

3. Results:

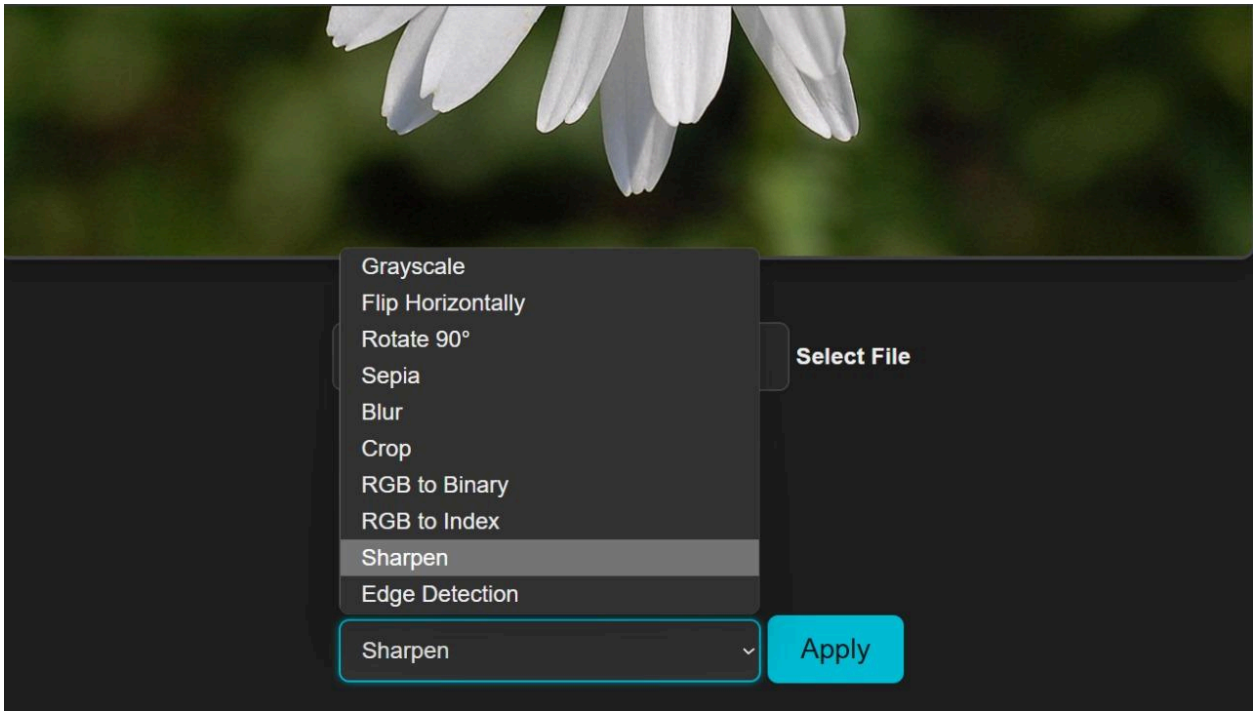
Output: Front page of website



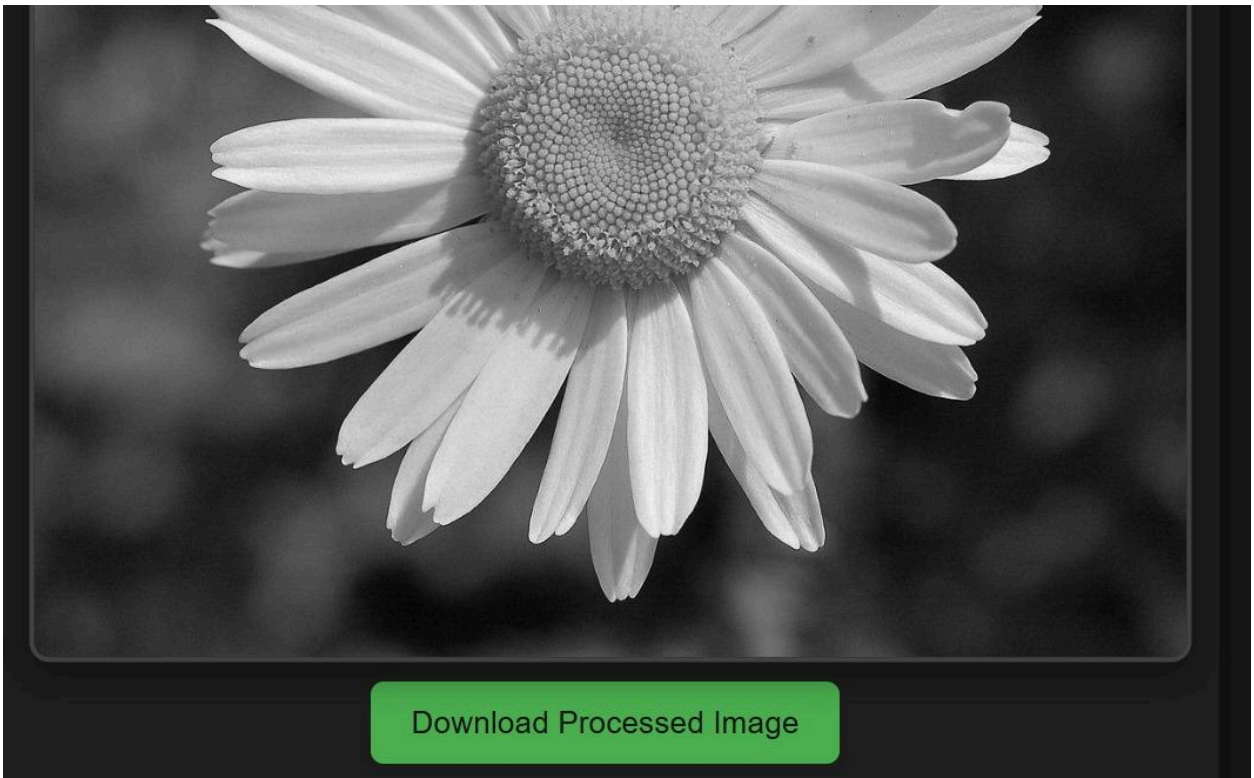
INPUT IMAGE:



Options Available:

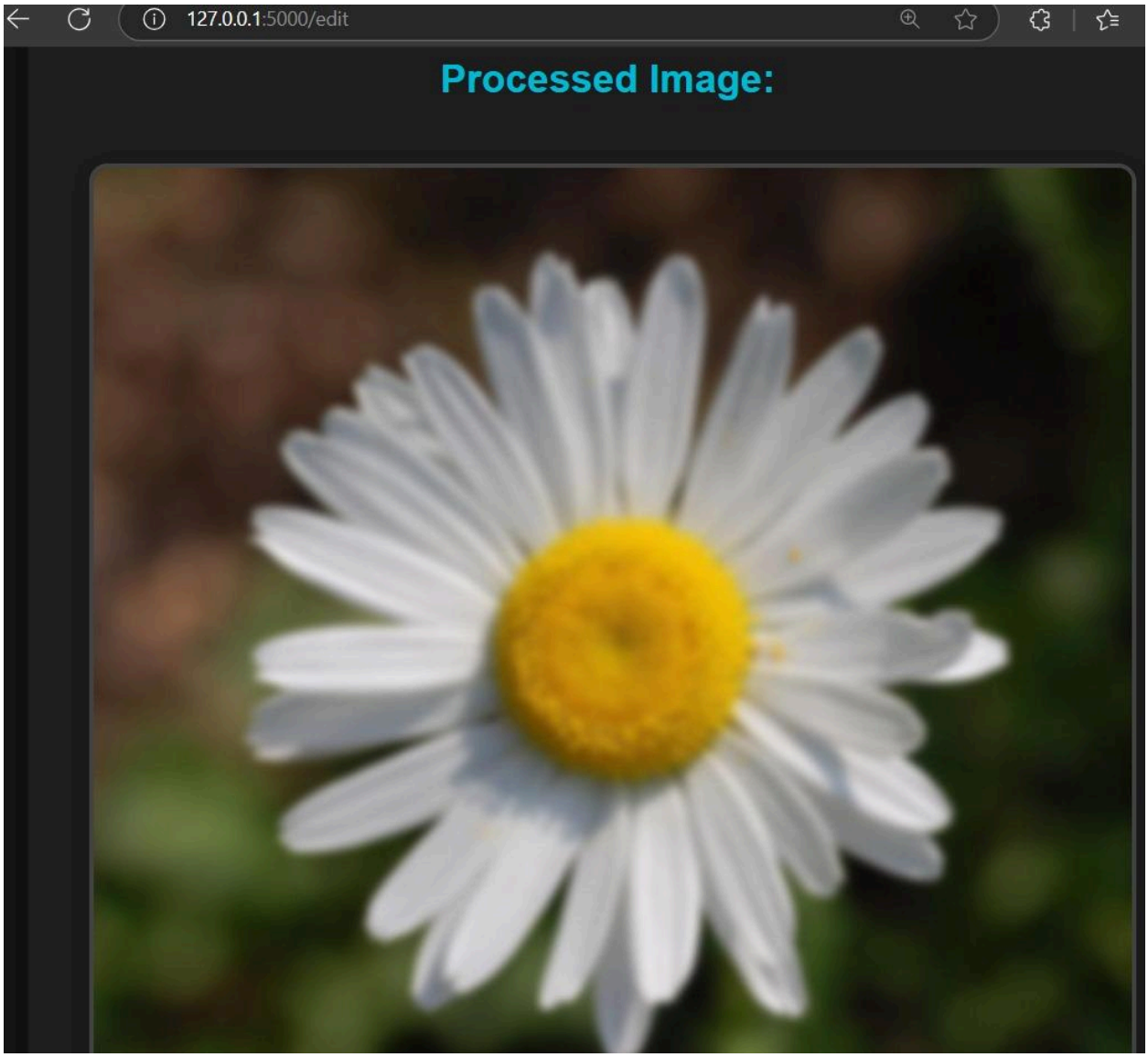


Grayscale Output:

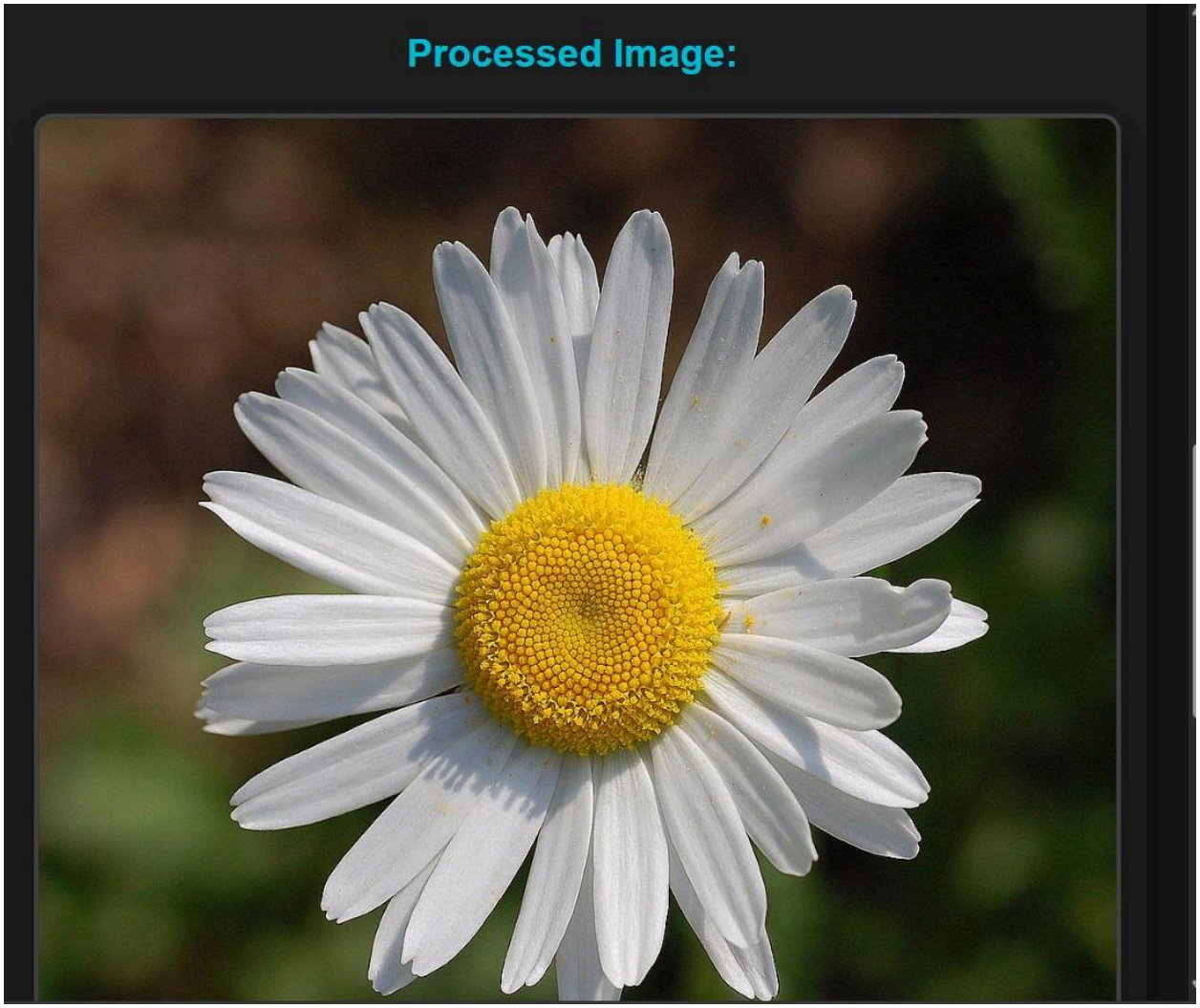


RGB to binary Output:

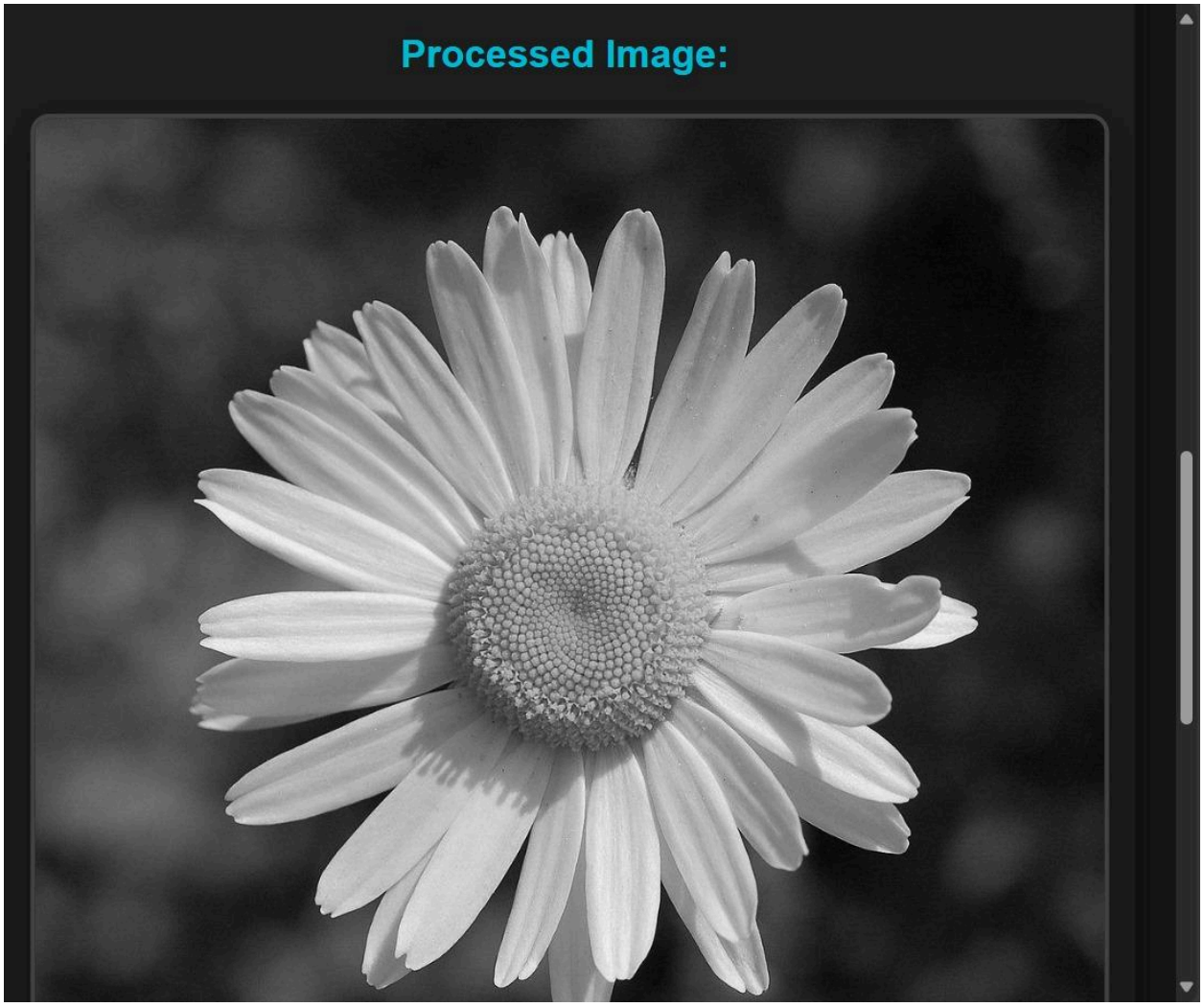
Blur Output:



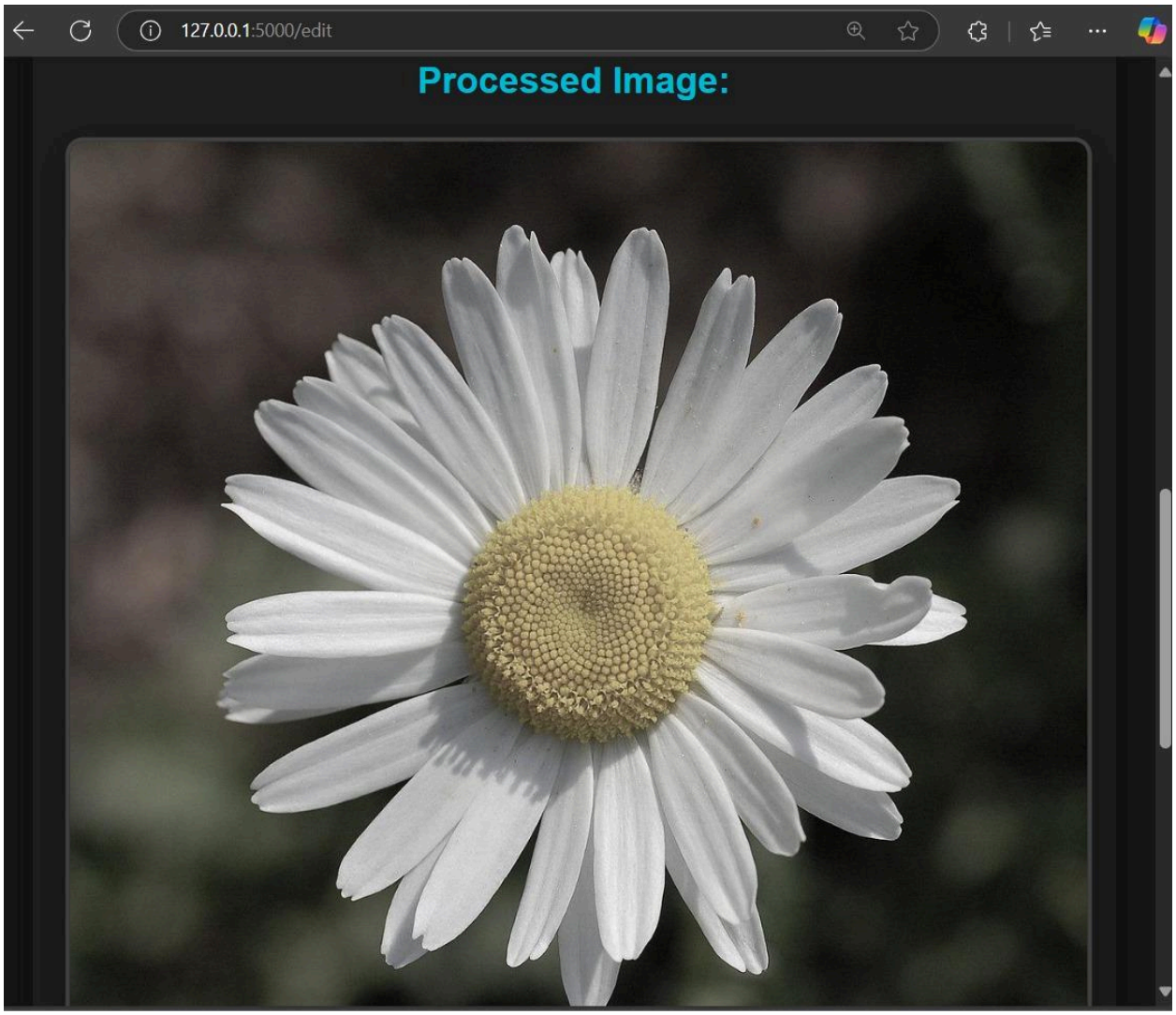
Sharpen Output:

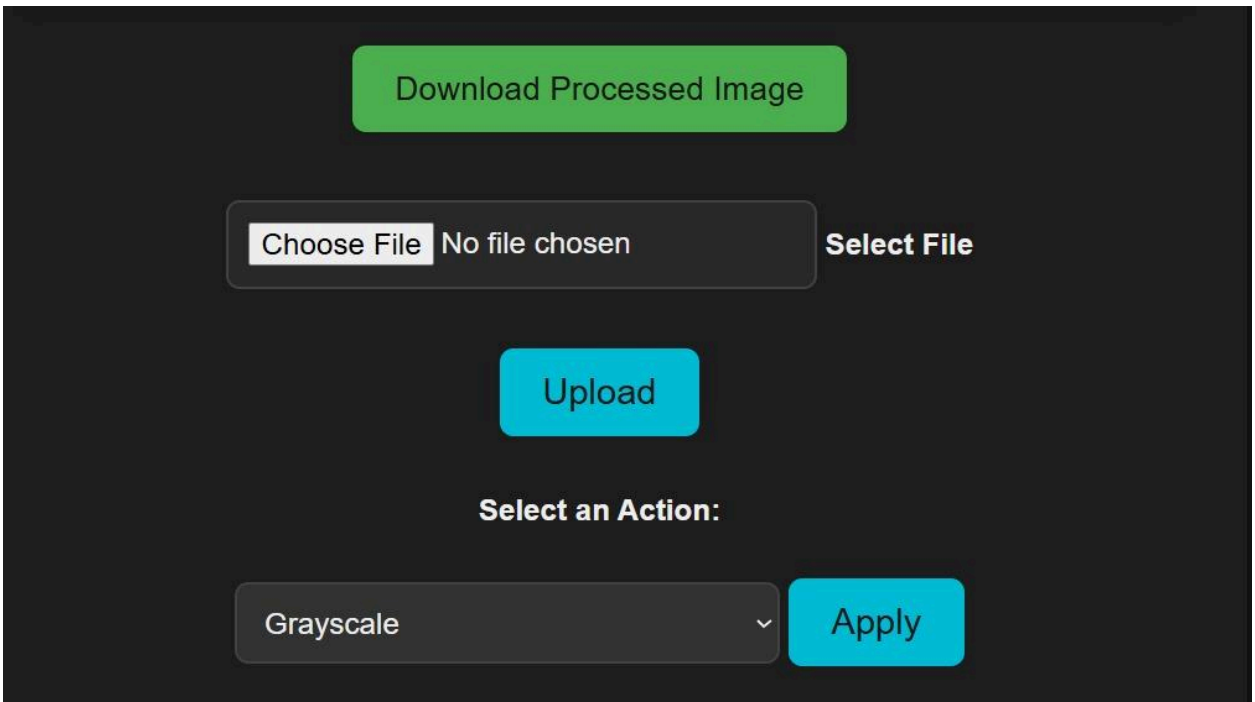


RGB to index Output:

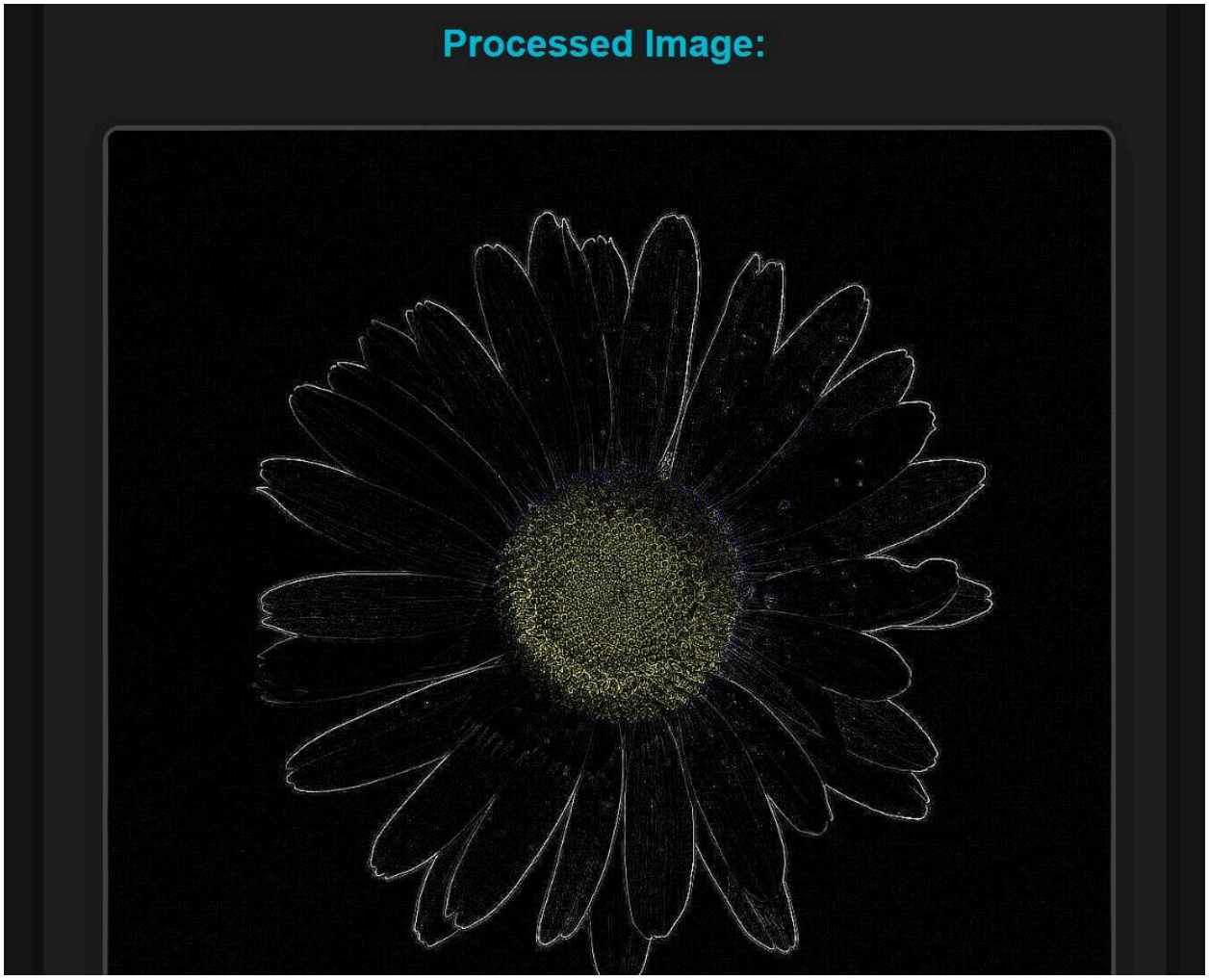


Sephia Output:





Edge detection Output:



TERMINAL CODE:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

PS C:\Users\shiva\OneDrive\Desktop\Ipcv_project> python app.py
PS C:\Users\shiva\OneDrive\Desktop\Ipcv_project> flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [22/Nov/2024 19:32:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2024 19:32:43] "GET /static/script.js HTTP/1.1" 304 -
127.0.0.1 - - [22/Nov/2024 19:32:43] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [22/Nov/2024 19:32:43] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [22/Nov/2024 19:33:28] "POST /upload HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2024 19:33:28] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [22/Nov/2024 19:33:28] "GET /static/uploads/flower.jpg HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2024 19:33:28] "GET /static/script.js HTTP/1.1" 304 -

Ln 5, Col 1  Spaces: 4  UTF-8  CRLF  {} Python
```

4.Conclusion :

EditMaster bridges the gap between simplicity and professional-grade image editing, empowering users with an intuitive interface and advanced features. By streamlining the editing process, it caters to diverse needs, from casual social media posts to complex graphic design projects, fostering creativity and enhancing productivity for everyone.

MADE BY : HARSHIT (**UI22EC28**), DHRUV KUMAR MEENA(**UI22EC22**), SHIVAM SAGAR(**UI22EC69**).