**HTML**

DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Photo Editing Tool</title>

  <link rel="stylesheet" href="s.css">

</head>

<body>

  <!-- Navigation Bar -->

  <nav>

    <ul>

      <li><a href="#photo-editor">Photo Editor</a></li>

      <li><a href="#features">Features</a></li>

    </ul>

  </nav>

  <!-- Photo Editor Section -->

  <section id="photo-editor">

    <h1>Photo Editing Tool</h1>

    <div class="editor-container">

      <input type="file" id="upload" />

      <canvas id="canvas"></canvas>

      <div class="controls">

        <button id="crop-btn">Crop</button>

        <button id="resize-btn">Resize</button>

        <button id="filter-btn">Apply Filter</button>

      </div>

    </div>

  </section>

  <!-- Features Section -->

  <section id="features">

    <h1>Features</h1>

    <ul>

      <li>Crop, Resize, and Apply Filters to Photos</li>

      <li>Responsive Design for All Devices</li>

      <li>Easy-to-use Interface</li>

    </ul>

  </section>

  <script src="script.js"></script>

</body>

</html>

**CSS**

/\* Global Styles \*/

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-image: linear-gradient(rgb(141, 180, 75), rgb(45, 45, 30), green);

  }

  h1 {

    color: #191717;

  }

  ul {

    list-style-type: none;

    padding: 0;

  }

  /\* Navbar Styles \*/

  nav {

    background-image: radial-gradient(rgb(15, 186, 239),rgb(59, 70, 59),rgb(38, 38, 46));

    padding: 10px 20px;

    position: sticky;

    top: 0;

  }

  nav ul {

    display: flex;

    justify-content: flex-start;

  }

  nav ul li {

    margin-right: 20px;

  }

  nav ul li a {

    color: rgb(240, 239, 248);

    text-decoration: none;

    font-size: 18px;

  }

  /\* Section Styles \*/

  section {

    padding: 20px;

    margin-top: 50px;

  }

  .editor-container {

    display: flex;

    flex-direction: column;

    align-items: center;

  }

  #canvas {

    border: 1px solid #c22222;

    margin: 20px 0;

  }

  .controls button {

    padding: 10px;

    margin: 5px;

    background-color: #ca0a91;

    color: rgb(250, 23, 23);

    border: none;

    cursor: pointer;

  }

  .controls button:hover {

    background-color: #bbda0b;

  }

  /\* Responsive Styles \*/

  @media (max-width: 768px) {

    nav ul {

      flex-direction: column;

      align-items: center;

    }

    .editor-container {

      width: 100%;

    }

    #canvas {

      width: 100%;

      max-width: 500px;

      height: auto;

    }

  }

**JS**

// Get elements

const uploadInput = document.getElementById("upload");

const canvas = document.getElementById("canvas");

const cropBtn = document.getElementById("crop-btn");

const resizeBtn = document.getElementById("resize-btn");

const filterBtn = document.getElementById("filter-btn");

const ctx = canvas.getContext("2d");

let img = new Image();

// Handle image upload

uploadInput.addEventListener("change", (e) => {

  const file = e.target.files[0];

  if (file) {

    const reader = new FileReader();

    reader.onload = () => {

      img.src = reader.result;

    };

    reader.readAsDataURL(file);

  }

});

// Handle image loading

img.onload = () => {

  canvas.width = img.width;

  canvas.height = img.height;

  ctx.drawImage(img, 0, 0);

};

// Crop image (simple rectangular crop)

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

  const croppedCanvas = document.createElement("canvas");

  const croppedCtx = croppedCanvas.getContext("2d");

  const cropWidth = img.width / 2; // Crop half of the width

  const cropHeight = img.height / 2; // Crop half of the height

  croppedCanvas.width = cropWidth;

  croppedCanvas.height = cropHeight;

  croppedCtx.drawImage(img, 0, 0, cropWidth, cropHeight, 0, 0, cropWidth, cropHeight);

  ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear canvas

  ctx.drawImage(croppedCanvas, 0, 0); // Draw cropped image

});

// Resize image (50% resize for demonstration)

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

  const resizedWidth = img.width \* 0.5;

  const resizedHeight = img.height \* 0.5;

  canvas.width = resizedWidth;

  canvas.height = resizedHeight;

  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(img, 0, 0, resizedWidth, resizedHeight);

});

// Apply a simple grayscale filter

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

  ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear canvas

  ctx.filter = "grayscale(100%)";

  ctx.drawImage(img, 0, 0, canvas.width, canvas.height);

  ctx.filter = "none"; // Reset filter for future actions

});