

Create the Frontend (HTML + JS)

This frontend will serve two main purposes:

- User Interface: Provide a way for the user to select an image file.
- AWS SDK Integration: Use the AWS SDK for JavaScript to upload the selected image to your S3 input bucket, leveraging the temporary credentials provided by Cognito Identity Pool. It will also attempt to display the processed image.

1. Create a File Named index.html

In your working directory, create a new file called index.html and paste the following code into it.

```
<!--
```

SETUP INSTRUCTIONS:

⚠️ Replace the following placeholders in the code below:

1. 'YOUR_AWS_REGION' — Replace with your AWS region (e.g., 'ap-south-1')
2. 'YOUR_COGNITO_IDENTITY_POOL_ID' — Replace with your AWS Cognito Identity Pool ID
3. 'YOUR_INPUT_BUCKET_NAME' — Replace with your S3 bucket name where users upload images
4. 'YOUR_OUTPUT_BUCKET_NAME' — Replace with your S3 bucket where resized images are stored

This file uploads an image to S3, waits for Lambda to resize it, and displays the original and resized image sizes along with preview.

```
-->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Upload Image to S3</title>
```

```

<script src="https://sdk.amazonaws.com/js/aws-sdk-2.1470.0.min.js"></script>
<style>
  body { font-family: Arial, sans-serif; padding: 20px; }
  img { max-width: 300px; margin: 10px 0; }
  #preview, #output { display: block; }
</style>
</head>
<body>
  <h2>Upload Image to S3</h2>
  <input type="file" id="fileInput" accept="image/jpeg, image/png"/>
  <button onclick="uploadFile()">Upload</button>
  <p id="status"></p>
  <p id="originalSize"></p>
  <p id="resizedSize"></p>
  <p id="resizedUrl"></p>

  <h3>Preview</h3>
  

  <h3>Processed Image</h3>
  

  <script>
    // AWS Config - Replace these values with your own
    AWS.config.region = 'YOUR_AWS_REGION'; // e.g., 'ap-south-1'
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'YOUR_COGNITO_IDENTITY_POOL_ID'
    });

    function uploadFile() {
      const fileInput = document.getElementById('fileInput');
      const file = fileInput.files[0];

      if (!file) {
        document.getElementById('status').innerText = 'Please choose a file.';
      }
    }
  </script>

```

```

    return;
}

// Show preview
const reader = new FileReader();
reader.onload = function(e) {
    const preview = document.getElementById('preview');
    preview.src = e.target.result;
    preview.style.display = 'block';
};
reader.readAsDataURL(file);

// Show original size
document.getElementById('originalSize').innerText = `Original Image Size:
${(file.size / 1024).toFixed(2)} KB`;

const s3 = new AWS.S3({
    apiVersion: '2006-03-01',
    params: { Bucket: 'YOUR_INPUT_BUCKET_NAME' }
});

const params = {
    Bucket: 'YOUR_INPUT_BUCKET_NAME',
    Key: file.name,
    Body: file,
    ContentType: file.type
};

document.getElementById('status').innerText = 'Uploading...';

s3.upload(params, function (err, data) {
    if (err) {
        console.log(err);
        document.getElementById('status').innerText = 'Error: ' + err.message;
    } else {

```

```

    console.log('Upload success', data.Location);
    document.getElementById('status').innerText = 'Upload success.
Processing...';

    const resizedFileName = 'resized-' + file.name;
    const outputUrl =
`https://YOUR_OUTPUT_BUCKET_NAME.s3.YOUR_AWS_REGION.amazonaws.com
/${resizedFileName}`;

    setTimeout(() => {
        const outputImg = document.getElementById('output');
        outputImg.src = outputUrl;
        outputImg.style.display = 'block';


        document.getElementById('resizedUrl').innerText = `Resized Image URL:
${outputUrl}`;

        fetch(outputUrl, { method: 'HEAD' })
            .then(response => {
                const contentLength = response.headers.get('Content-Length');
                if (contentLength) {
                    const kb = (parseInt(contentLength) / 1024).toFixed(2);
                    document.getElementById('resizedSize').innerText = `Resized Image
Size: ${kb} KB`;
                } else {
                    document.getElementById('resizedSize').innerText = 'Could not fetch
resized image size.';
                }

                document.getElementById('status').innerText = '✅ Image resized
successfully.';
            })
            .catch(error => {
                console.log('HEAD error:', error);
            })
    });

```

```

        document.getElementById('resizedSize').innerText = 'Could not fetch
resized image size.';
        document.getElementById('status').innerText = '  Image resized
successfully.';
    });

    }, 5000); // wait for Lambda to process
}
});
}
</script>
</body>
</html>

```

How to Use This Frontend:

1. Save the file: Save the code above as index.html in a folder on your computer.
2. Open in browser: Simply open the index.html file with your web browser (e.g., by double-clicking it).
3. Select a file: Click "Choose File" and select a .jpg or .png image.
4. Upload: Click the "Upload Image" button.

What to expect:

- The "Preview Original Image" section will show the image you selected.
- The status message will update as the file uploads and then as it waits for Lambda processing.
- After a short delay (controlled by the setTimeout in the JavaScript, adjusted to 7 seconds here), the "Processed Image" section should display the resized image from your output S3 bucket, along with its new size and URL.
- If the processed image doesn't appear, check your Lambda function's CloudWatch logs for any errors. Also, ensure your S3 output bucket has appropriate permissions (e.g., public read if you want the browser to access it directly, or consider presigned URLs for private buckets in a production setting).